DeepRad: Predictive Energy Dynamics for Building Facades

Brandon Cuffy
Georgia Institute of Technology
College of Computing
801 Atlantic Drive
bcuffy3@gatech.edu

Saeran
Vasanthakumar
Georgia Institute of Technology
College of Computing
801 Atlantic Drive
svasanth6@gatech.edu

Abstract

We develop a deep learning framework to predict the surface radiation dynamics of residential building facades through the implementation of an autoencoder-based transfer learning framework, an important, and time-consuming step in current building energy simulation workflows. Our framework uses incident surface solar radiation prediction as a source task for a second, more complex target surface interior solar radiation prediction. We theorize that we can isolate the well-known successive of deep learning models at projective transformations to conceptually decompose radiation prediction as an exterior and interior component. Autoencoders are proposed as an effective means of learning inherent facade features and to then reconstruct new images. Our experimentation demonstrated the importance of capturing the nonlinear nature of the source task through the use of sigmoid activation in the decoder. This yielded promising results after parameter tuning, with resulting images matching the target solar radiation simulation values. However, we did achieve accurate results on the target task of predicting interior radiation, despite manual and automated model tuning. Additional refinement is needed to achieve higher fidelity, and generalizable results. Despite this, and the lack of success of our target task, we believe we have promising results that suggest a repository of simulated solar analysis data can be leveraged in the future to predict incident, and interior-transmitted solar radiation on buildings without the need for simulation.

1. Introduction

Deep convolutional neural networks (CNNs) achieve great performance in real world computer visual recognition and prediction problems. This is typically attributed, in part, to the ability of deeply nested, convolutional and nonlinear activation functions to successively extract image statistics that correspond to the transformation and visual perception of 3D geometries, and other semantic information. This process takes advantage of the ability to generalize local correlation at the pixel level towards the 2D projection of 3D transformations and lighting at the object level [Alammar et al.].

These properties are in turn a function of the way the geometry and material of these 3D objects reflect the visible light range of the electromagnetic (EM) spectrum. This suggests that the success of CNNs in computer vision can be translated to adjacent problem domains that are a function of the human-perceptible EM spectrum, even if they aren't strictly computer vision tasks. One such example would be to classify and predict thermodynamic processes that are functions of the radiative heat transfer, which like light is absorbed, reflected, or transmitted by some constant fraction, and, like light is emitted in all directions associated with a hemispherical surface. Radiative heat transfer can be mapped to the image domain (i.e through false color maps), but unlike light, it's flux is firmly governed by temperature difference relative to geometric surfaces, openings and volumes.

Our research proposal is to reframe the energy impact of building envelopes as a computer vision problem, by using CNNs to accurately predict radiative heat transfer. Specifically, we address limitations in existing state-of-the-art methods to predict building energy performance by building multiple CNNs within a transfer learning framework that encodes specific material and geometric priors relevant for radiative heat transfer. We believe our attempt to leverage the success of computer vision-oriented CNNs towards an adjacent, non-visual problem domain acts as a useful application of transfer learning that can improve the existing understanding of inductive biases in CNNs. Specifically, by emphasizing different properties of the somewhat counter-intuitive shape-texture relationship that drives CNN success (Geirhos et al. 2019), we can develop intuition about relative success of overlaying thermal properties in the image domain to drive inference. Despite the similarities, we predict some significant challenges in translating visual prediction to radiation prediction in data generation. Specifically, unlike the proliferation of labelled, and unlabelled photographic data, radiation imagery will have to be manually generated in a way that efficiently captures the three-dimensional representation of building geometry.

2. Background Motivation

The use of machine learning to predict building energy consumption has typically been limited to models that coarsely summarize geometry through statistical descriptions (Aijazi, 2017), due to the difficulty in accounting for the highly non-linear interaction between radiation and building geometry. Such models thus generalize poorly on the kind of geometrically complex structures typically used in the architecture and engineering domain.

Creating a model that can predict geometrically complex structures would therefore improve on state-of-the-art methods used in the architecture and engineering domain, which currently relies on time-consuming, complex numerical simulations to simulate building energy consumption. This is then compounded by the need to perform multi objective optimizations that balance efficient building energy consumption against other design constraints. This is a non-convex optimization problem and thus requires multiple simulation iterations through stochastic optimization methods, most typically genetic algorithms. Speeding up this process is important given the disproportional impact of buildings on the greenhouse gas emissions and the increasing regulatory demand for predicting and optimizing building energy consumption.

3. Approach

Our transfer learning framework consists of two autoencoder networks representing source and target radiation prediction tasks, respectively. We chose transfer learning as an overall framework because the source task (predicting incident exterior surface radiation) is a more general version of the target task (predicting interior surface radiation gain). This is because interior radiation is a function of not only the radiation transmitted from the exterior (which the source task will be trained to predict) but also the interaction of the radiation bouncing and scattering within closely set interior surfaces.

The task predictors themselves were modeled with autoencoders. Autoencoders were chosen as our project architecture because we chose to encode the 3d building geometry into images as an efficient lower dimensional representation. This decision is another reason why the target task is more difficult then the source task: there is more visual (and radiative) obstruction from interior structures along the depth of the interior which ideally needs to be interpreted by the neural network as a perspective transformation of 3d geometry.

Dataset To predict surface solar radiation from building geometry, we needed to generate building geometry data, simulate it's energy performance, and then capture orthographic views of the building geometry, with and without radiation visualized as a color map for feature and label data.

We generated the building geometry from the the CubiCasa5K (Kalervo et al., 2019) dataset, which contains raster and SVG data of 5000 residential and office buildings. We built a custom pipeline using to generate building energy models from this dataset (available in the supplementary material as ”writefloorplanjson.py” and ”image2vec.py”). The overall workflow was to clean and extract polygon contours from the raster floorplan images using OpenCV, using the SVG files to scale the buildings, and then applying a Gaussian Mixture Model from scikit-learn to identify an orthogonal grid to snap building polylines in a watertight fashion (a requirement for energy modeling). The watertight polygons were then extruded by a constant height twice to represent a two-story sample building.

The geometries were then annotated with building energy, and environmental data, using the Ladybug Tools Python library from Python (Ladybug Tools, 2021) in the 3D CAD modeler Rhinocerous3D. Each building was represented as a typical midrise residential apartment building in Philadelphia. Building construction was modelled as a brick structure typical to rowhouses in Philadelphia, and windows were placed on each facade. Window size was determined from a uniform distribution between 0.1 and 0.9, with a solar transmittance value of 0.3. This was to ensure a reasonable distribution of interior surface radiation was present in the training data.

An energy simulation was then performed for each model, using the climate data of Philadelphia. The analysis results and geometric information was then post-processed to produce orthographic diagrams of the building facades using Pincam (Vasanthakumar, 2021) a personal library developed by one of the authors for simulating light-weight camera views of building energy geometries. Figures 1 and 2 illustrate the final genereated input and output data for the source and target tasks.

For both tasks the input data consists of the exterior geometry of the building colored by material type. The target task also includes window geometries as a separate channel. These were used as a single channel or concatenated channel for the first convolutional layer of the respective autoencoders. The outputs are the same geometries colored by solar radiation data extracted from the simulation analysis. Each geometry consists of the same building at five different orientations (rotated by 45 degrees) at a pitch of 15 degrees.

Initially we modeled the input and output geometries with a zero degree pitch at only the cardinal orientations, along with a depth map of each surface. However we found that the model was not able to effectively extract low-level descriptive features of each facade, even with a depth map channel. The resulting ”orthographic” perspective was a compromise between efficiently encoding planar relationships of the building, and maintaing a simple planar represention from which the entire facade can be seen and used for analysis. This enabled the model to better understand and encode solar radiation data as it relates to spatially explicit areas of the building geometry, possibly because of the visual redundancy introduced by the 45 degree rotation, which provides areas of ”pixel” overlap for the network to better identify the geometric relationship.

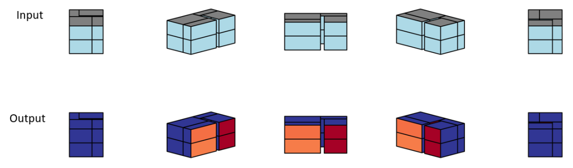


Figure 1. Source Task Input and Output Data

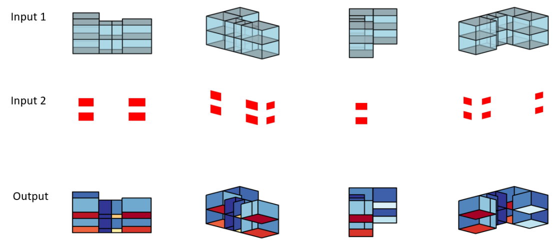


Figure 2. Target Task Input and Output Data

Model The primary task of the network is to consume images of the facade geometry, with no windows, and produce cumulative incident solar radiation and illuminance false color maps per facade, along with vector information about the average direction of cumulative radiation. Since reconstruction is critical for this cnn, an autoencoder architecture was chosen. Autoencoders are a self-supervised neural network traditionally used for dimensionality reduction to learn low level features in an image [2]. They are able to map an input x to an output, known as a reconstruction, via an learned representation of salient features in the input image. [1] The encoder $g(\phi)$ and the decoder $f(\theta)$ have two learnable parameters ϕ and θ that enable the network to reconstruct the target image [3]. The learned parameters are represented in the model as $\mathbf{z} = g_\phi(\mathbf{x})$ such that when passed through the decoder, the reconstructed target image is $\mathbf{x}' = f_\theta(g_\phi(\mathbf{x}))$.Putting this altogether, the self-supervised metric of the autoencoder can be describe as:

$$L_{\mathrm{AE}}(\theta, \phi) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}^{(i)} - f_\theta(g_\phi(\mathbf{x}^{(i)})))^2$$

Figure 3. Autoencoder Impage Comparison Metric

The features to be learned in this case are the three-dimensional characteristics of a building facade. Once these low level features become embeddings through the encoder, the model then uses a decoder to reconstruct the, in this case, the solar radiation target image. Futhermore, this allows the output of the model to be compared pixel-to-pixel with the target image. Mean squared error(MSE) is used to calculate loss between image reconstructions and the target image, Figure 4. The aim is to enable the model to represent geometric compositions and how they affect solar radiation on a given facade area.

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = (x_n - y_n)^2$$

Figure 4. Mean Squared Error

The first network is implemented with python’s PyTorch deep learning library. Part of the architecture is based on experiments explored in APS260 Artifical Intelligence Fundamentals [4]. Quantity of input/output channels, number of layers, and activation functions were all modified, tested, and adapted from the original code base. In addition, our network proposes the reconstruction of a novel image introduced to the encoder, rather than the traditional reconstruction of the input image. The convolutional network has two main components, an encoder and decoder, Figure 5. Then

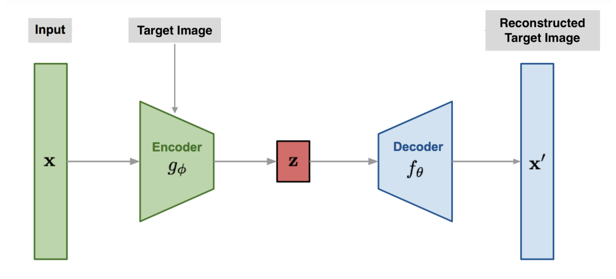


Figure 5. Autoencoder Diagram [3]

encoder is a composed of three convolutional layers with ReLu activation after each, except the last cnn layer. The decoder is composed of three, two-dimensional transposed convloutional layers with ReLu activation. Transposed convolutional layers are used to reconstruct he target image from the low level features output from the encoder layers. Because the problem outlined in the paper is not linear, Sigmoid activation is implemented as the last layer of the decoder to account for non-linearities in the model.

The second cnn, implmented in a transfer learning framework, consumes images of solar radiation illuminance false color maps, and thermal material information as additional information channels (i.e. for capturing reflectivity, conductance), along with a building floor plan, and produce a building energy consumption prediction. Furthermore, a then perform a forward pass along a single image, and then perform stochastic gradient ascent to generate window geometries that minimize building energy consumption.

4. Experiments and Results

The general strategy for the optimization of the model was to manually tune the autoencoder workflow to determine broad model design features (i.e. determining if the second network should freeze the encoder layers of the first network or not), and then using the Ray Tune library (Moritz et al) for automated hyperparameter tuning. This section summarizes the manual experimentation to identify the initial autoencoder, hyperparameter tuning of the Adam optimizer, manual tweaking of the non-linear layers, and the hyperparameter tuning of feature and kernel size.

Initial Design Experimentation First, the initial Autoencoder structure was manually tuned to produce a legible image. This consisted of: first testing number of output channels for each cnn layer ranging from 32, 64, 128 and 256. However, increasing the depth of the last convolutional layer in the encoder did not significantly improve accuracy of the outputs. We use 64 output channels for the final implementation. Two known limitations of this initial model were the lack of max pooling, and nonlinear activation layers. Maxpooling was intended to be implemented later on. For this reason, increasing the output channels by increasing the receptive field, may have yielded sub-optimal results because the network is not using pooling layers, which could help summarize increased features identified by the additional receptive fields. The second limitation was the lack of a non-linear activation function. This wasn't implemented because the outputs were normalized between 0 and 1, which we reasoned would not be aided by the exponential scaling of the sigmoid layer. ReLU activation layesr were used instead to act as a more stable nonlinear activation layer in the model. This initial model was then used as a baseline for subsequent experiments. A typical learning curve, and associated outputs from the baseline model is illustrated below for 30 epochs with a batch size of 5.

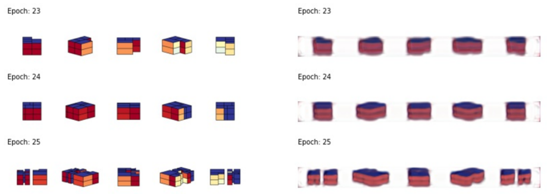


Figure 6. Baseline Prediction

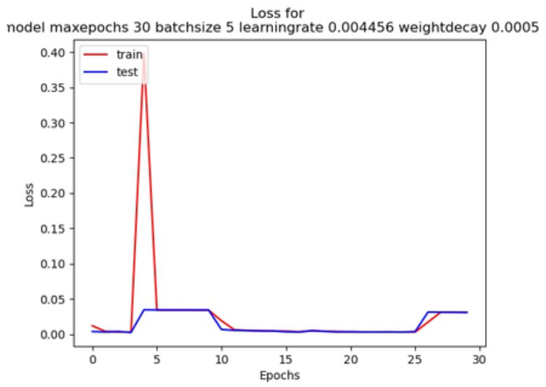


Figure 7. Baseline Model Learning Curve

As can be seen, while the baseline model generally had a test MSE loss between 0.005 and 0.003 which produced a legible image, the learning curves for this model didn't descend smoothly, and thus didn't lend itself to improvement. Our first attempt to improve the model learning was by tuning the learning rate and weight decay for the Adam optimizer. We use the Ray Tune library to run multiple trials where parameter values for each were randomly, and logarithmically sampled from a uniform distribution over: $U[1 \cdot 10^4, 1 \cdot 10^{-4}]$ and $U[1 \cdot 10^{-5}, 1 \cdot 10^{-1}]$, for 30 epochs with a batch size of 5. A sample from these trials are illustrated in Figure 8 and 9.

While the gradient descent of the model improved over the baseline, reaching at it's best a test loss of 0.002, all descent is concentrated in the first few epochs, and doesn't improve beyond that. For this reason the image predictions did not significantly improve over the baseline.

Testing non-linear Activation To improve the slope, we manually testing the addition of a Sigmoid, Softmax and TanH activation layer at the final step of the autoencoder. We found the Sigmoid layer had the best impact, immediately improving the slope of the learning curves (Figure

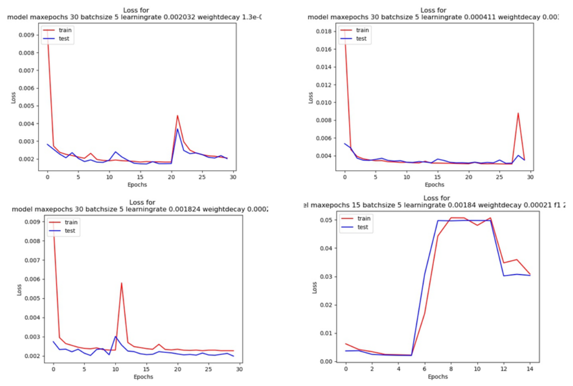
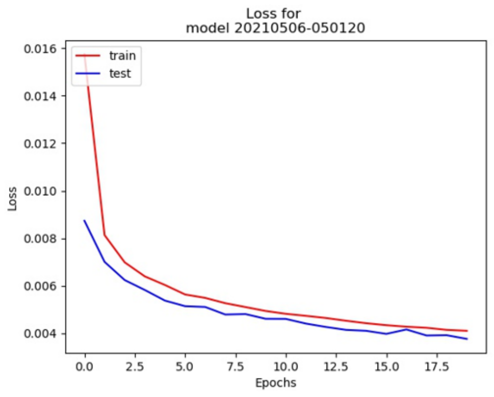


Figure 8. Learning Curves

11) but as can be seen by the learning curves, didn't converge to a solution. However, because it'd didn't plateau, this model leant itself to hyperparameter optimization, and we were quickly able to identify optimal hyperparameters for the optimizer (learning rate, weight decay) as well as the autoencoder (kernels and feature depths) with Ray Tune. The results are illustrated below.



last test_loss: [0.00376571]

Figure 9. Learning Curve with Sigmoid Activation after optimization

This model performed the best, and had a promising learning curve that indicated further optmization is possible.

(10 points) How did you measure success? What experiments were used? What were the results, both quantitative and qualitative? Did you succeed? Did you fail? Why? Justify your reasons with arguments supported by evidence and data.

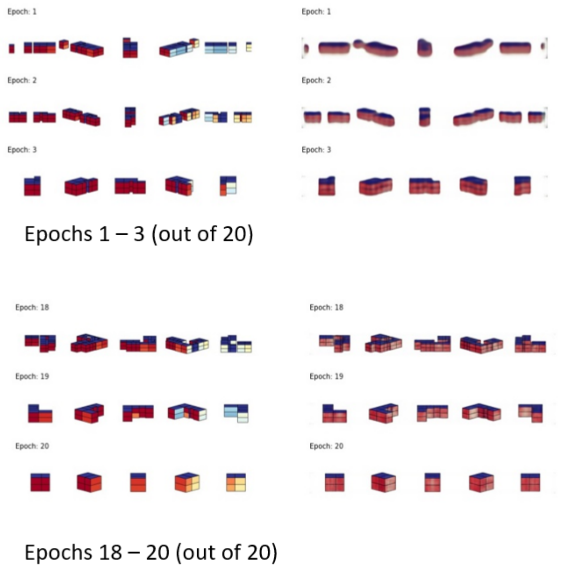


Figure 10. Prediction with Sigmoid Activation after optimization

Second Encoder The optimized first autoencoder, which successfully predicted a source target of exterior surface radiation was then used to aid the prediction of a target task in a more specific domain: interior surface radiation. This was the second component of our original proposal, to transfer the latent knowledge from the general surface radiation prediction model to aid interior surface radiation prediction from a model of the building facade. This second component is quite a bit more difficult then the first component since it entails predicting the amount of solar radiation that is transmitted and scattered through glass, and then reradiated and scattered amongst interior surfaces. Our optimization strategy for this component of the transfer learning system was the same as the first component: we manually tuned larger design features, and use Ray Tune for hyperparameter tuning.

The initial results of the transfer learning was not successful. We tested two methods of transfer learning: using the pretrained source as the source of initial weights for a typical training and testing iteration, and using the pretrained source with its encoder frozen, so that the weights were only updated in the decoder structure. The learning curves, and predicted images from this test are presented in the figure below. In general there wasn't a huge difference between the two approaches in terms of average loss, although there were differences in the behaviour of the models. Specifically, freezing the encoder resulted in a more unstable test curve, with many osciliattions, indicating the model needed to be regularized. The unfrozen model showed a more stable descent, although the test data performed better then the training data, indicating they were not distributed evenly between the two. We implemented a random selection of test and training data, so the limitation was with the lack of data (in this instance, around 1200 images).

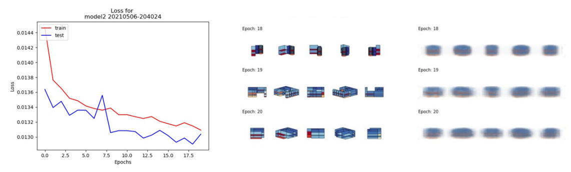


Figure 11. Transfer learning with original model.

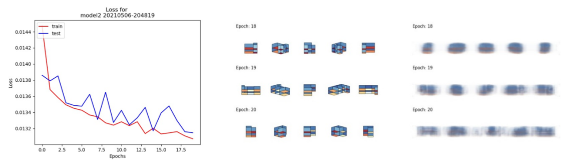


Figure 12. Transfer learning with Encoder Frozen

5. Conclusion

Our project indicated partial success, the initial incident solar radiation prediction performed well on test data, but the second, more difficult task of interior solar radiation did not. There is still additional refinement needed to achieve higher fidelity result for this portion likely due to the increased complexity of the target domain, and less time we had to tune the hyperparameter o fthe project. In general we feel the autoencoder structure proved to be a successful deep learning architecture for geometric radiation prediction, and thus feel with further time the second component of DeepRad would work, based on the success we had with the first component.

6. Conclusion

7. Work Cited List

You are welcome to introduce additional sections or subsections, if required, to address the following questions in detail.

(5 points) Appropriate use of figures / tables / visualizations. Are the ideas presented with appropriate illustration? Are the results presented clearly; are the important differences illustrated?

(5 points) Overall clarity. Is the manuscript self-contained? Can a peer who has also taken Deep Learning understand all of the points addressed above? Is sufficient detail provided?

(5 points) Finally, points will be distributed based on your understanding of how your project relates to Deep Learning. Here are some questions to think about:

What was the structure of your problem? How did the structure of your model reflect the structure of your problem?

What parts of your model had learned parameters (e.g., convolution layers) and what parts did not (e.g., post-processing classifier probabilities into decisions)?

What representations of input and output did the neural network expect? How was the data pre/post-processed? What was the loss function?

Did the model overfit? How well did the approach generalize?

What hyperparameters did the model have? How were they chosen? How did they affect performance? What optimizer was used?

What Deep Learning framework did you use?

What existing code or models did you start with and what did those starting points provide?

Briefly discuss potential future work that the research community could focus on to make improvements in the direction of your project's topic.

8. Work Division

Please add a section on the delegation of work among team members at the end of the report, in the form of a table and paragraph description. This and references do **NOT** count towards your page limit. An example has been provided in Table 1.

Student Name	Contributed Aspects	Details
Brandon Cuffy	Data Processing/Generation, Implementation, Writing	Helped pre-process and generate dataset for initial 2D facade/solar Radiation images. Implmented the autoencoder convolutional netwrok. Helped write report
Saeran Vasanthakumar	Dataset Procession/Generation, Implementation, Writing	Ran solar incident radiation and view extraction script to create the base dataset. Implemented transfer learning framework. Helped write report

Table 1. Contributions of team members.

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning, 2016. Book in preparation for MIT Press. 3

[2] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019. 3

[3] Lilian Weng. From autoencoder to beta-vae. https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html. Accessed: 2021-04-30. 3

[4] Lisa Zhang. Aps360 artificial intelligence fundamentals. https://www.cs.toronto.edu/~lczhang/360/. Accessed: 2021-04-30. 3