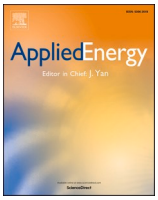
Applied Energy 360 (2024) 122696

Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apenergy



A U-net convolutional neural network deep learning model application for identification of energy loss in infrared thermographic images

David Gertsvolf, Miljana Horvat, Danesh Aslam, April Khademi, Umberto Berardi *

Toronto Metropolitan University, 350 Victoria Street, Toronto, Ontario M5B 2K3, Canada

HIGHLIGHTS

- Deep Learning potential in the construction sector is explored.
- Infrared imaging processing and interpretation is combined with the potential of AI.
- Convolution Neural Networks are used for finding deficiencies in building envelopes.
- Biomedical imaging techniques are applied to aerial infrared images of buildings.

ARTICLE INFO

Keywords:
Infrared thermography
Building envelope assessment
Deep learning
Convolution neural network
U-NET

ABSTRACT

The possibility of obtaining large data set of infrared images during building and urban envelope surveys require the development of fast and effective ways to process their content. This study presents a novel U-NET convolution neural network (CNN) deep learning (DL) model for the identification of envelope deficiencies on a data set of infrared (IR) thermographic images of building envelopes. A data set of images acquired with an unmanned aerial vehicle (UAV) were used with supplementary segmentation masks created for appropriate U-NET modelling application. This data preparation process is presented followed by an in-depth review of the CNN architecture used for the segmentation process. The Python3 code developed for this study is simplified for easier application by non-data-science researchers. The results of this research show high accuracy. However, large data set are needed to better train the CNN-DL model.

1. Introduction

The building sector accounts for roughly 30–40% of global energy usage which in turn contributes to over a third of the world's greenhouse gas emissions [1]. The worldwide growing awareness of building energy consumption and the resulting negative environmental impact have urged policymakers to focus on energy conservation [2].

Building energy performance is mainly impacted by defects such as poor envelope design, and construction, particularly dominated by air leakages and inadequate insulation. With energy loss through the building envelope being the leading contributor to poor building energy efficiency, the identification of inadequately envelope performance is required prior to the implementation of appropriate rehabilitation actions [3]. While intrusive building envelope investigations are costly and time consuming, infrared (IR) thermography has become a common practice for default detection. In addition, advancements in infrared and digital imaging technologies coupled with the substantial price drop of IR cameras has increased the use of this non-intrusive inspection method [4].

The use of unmanned aerial vehicles (UAVs) has recently become a contributor to the thermographic analysis granting more accessible and more accurate imaging collecting methods. As the collection of infrared thermal images has evolved, the evaluation techniques of the imageries still require extensive developments. Current methods typically require an expert in building science and thermography to manually and individually evaluate the imagery and develop qualitative conclusions based on thermal patterns and knowledge of construction methods and materials in the assembly. This technique is inefficient as it is quite time consuming; in addition, this process is increasingly augmented when examining high-rise building envelopes as they require a considerably larger number of images.

Artificial intelligence (AI) and more specifically, machine learning (ML), is an intelligence system performed through machines where with

* Corresponding author.
E-mail address: uberardi@torontomu.ca (U. Berardi).

https://doi.org/10.1016/j.apenergy.2024.122696
Received 16 October 2023; Received in revised form 9 January 2024; Accepted 18 January 2024
Available online 15 February 2024

Abbreviations

Acronym	Definitions
CNN	Convolution Neural Network
DL	Deep Learning
IR	Infrared
ASTM	American Society for Testing and Materials
ISO	International Organization for Standardization
NMS	Canadian National Master Construction Standard
SCC	Standards Council of Canada
UAV	Unmanned Aerial Vehicle
GHG	Green House Gas
AI	Artificial Intelligence
ML	Machine Learning
ANN	Artificial Neural Network
FPA	Focal Plane Array
SVM	Support Vector Machine
KNN	k-Nearest Neighbour
NB	Naïve Bayes
DT	Decision Trees
RF	Random Forest
ROI	Regions of Interest
AEC	Architecture, Engineering and Construction
RGB	Red, Green, Blue
GPU	Compute Engine backend
1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional

Acronym	Definitions
BCE	Binary Cross Entropy
MSE	Mean Squared Error
SCC	Sparse Categorical Cross-Entropy

Symbol	Definitions
y	Normalized greyscale data
x	Original greyscale data
α	alpha
β	beta
TI	Tversky Index
TP	True Positive
FN	False Negative
FP	False Positive

minimal human intervention and supplied data, patterns and decisions can be developed. ML is able to analyze extensive amounts of data through the construction of automated analytical models. Depending on the type of collected information, as well as the intended objectives, there are various approaches and ML algorithms that can be employed. Deep Learning (DL) is a subset of ML that structures algorithms in layers to create an Artificial Neural Network (ANN) that can learn and make intelligent decisions on its own. DL uses structures similar to neurons to perform its learning system. A type of DL model is a Convolution Neural Network (CNN), that uses algorithms specifically designed for image processing. This DL model has become a key player in powering computer vision and has become a powerful tool in various fields including, biomedical engineering, self-driving automation and many others. With limited studies showing CNN applications in building envelope IR assessments, this research aims to answer the following question if CNN DL analytical models be applied to IR exterior building envelope images.

This study presents a development of a novel U-NET CNN, DL model, developed in a Python environment for the identification of deficient areas resulting in potential energy loss on a data set of IR thermographic images of building envelopes. First, a comprehensive review of existing state of the art building envelope assessment methods, with the use of thermography, ML and UAVs will be discussed in order to grasp an understanding of possibilities and limitations in this field. Following, the U-NET model that was programmed and adapted for assistance in the review process of building envelope IR images is presented. The aim is to segment and accurately identify energy loss on any given thermographic exterior wall system. The methodology consists of the image acquisition and data preparation procedures followed by a breakdown of the U-NET structure and development. The results section is broken down into two portions. First the development of the model throughout the research processed is presented for non-data-science personnel, followed by the accuracy and performance of the results.

2. Literature review

Building envelope is the physical separator between the conditioned and unconditioned environment and controls the air, water, heat, light, and noise transfer. When the air control and thermal control of an envelope is compromised, unwanted energy loss occurs through the envelope which increases the energy demand of a building. There are many factors that may affect the condition of a building envelope system starting from design to life cycle loads.

Infrared thermography (IR) has expanded as a mean of building energy auditing since the early 2000s, when portable IR cameras became affordable and available. The non-destructive testing method functions by producing a sequence of two-dimensional, temperature distributed, images. The IR camera captures the radiation emitted by objects at temperatures above absolute zero which are then converted to digital images that can be viewed by the human eye. IR covers a portion of the electromagnetic spectrum from approximately 0.9 to 14 μm. An IR detector cooling digitization device is a focal plane array (FPA) of micrometer size pixels made of various materials sensitive to IR wavelengths. The FPA detects photons and generates an electrical charge in relation to the number of photons detected at each pixel which is then measured, digitized, and used to construct a temperature image. These images are displayed through a colour palette.

Over the past two decades, IR thermography has evolved into a common building energy auditing tactic, both qualitative and quantitative. The qualitative testing approach identifies anomalies in the building envelope without defining temperature values and without translating to energy loss metrics. Qualitative IR assessments include: thermal characterization of walls, glazing and windows; thermal bridging and excessive heat loss areas detection; thermal insulation examination; air leakage inspection; moisture and water detection; HVAC and electrical systems characterization; indoor temperature measurements for human comfort assessment.

Quantitative analysis on the other hand, is used to quantify the thermal performance of a building envelope by presenting thermal transmittance values (U-values) based on the known interior and exterior temperatures. These calculations are performed on top of the IR images and require extensive building information. Quantitative building auditing methods include: determination of the percentage of areas with thermal anomalies; insulation levels detection; U-value measurements; dynamic characterization of walls, and moisture content determination [5].

IR analysis can be passive or active. Passive analysis measure the temperature difference under normal in-situ conditions while active thermography requires exerting an external source of heat on a wall element to induce an effect within a material that will emphasize any defects within the system [4,5,6]. Active analysis methods that incorporate thermal stimulation are typically used in laboratory testing, historic buildings or product characterization.

Fig. 1. Various uses of IR with UAVs: (a) LiDAR and 3D point cloud of Washington DC [11]; (b) Seaports are seen using thermography technology with aerial drone cameras [12]; (c) Thermal scanning of new multi-story buildings [13].

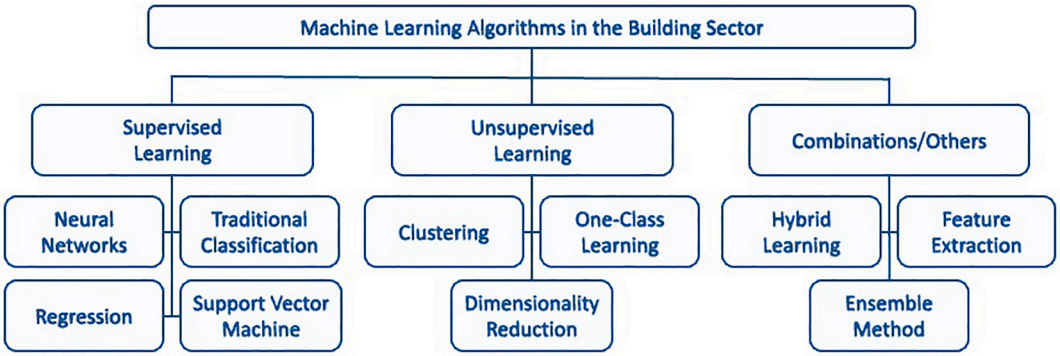


Fig. 2. Machine learning algorithms in the building sector.

Originally, IR imaging began with the use of handheld photographic cameras which was a limitation for the capturability and accessibility of entire building envelopes. The use of UAVs, i.e. drones, has pushed the limits of data capturing in the building fault detection field, enabling IR imaging on otherwise hard to reach areas of building envelopes. The majority of commercial photogrammetry and remote sensing (PaRS) is performed with UAV cameras. Supported software for flight planning, flight navigation, guidance and control, and data post-processing exists [7]. IR imaging has also helped with the development of 3D model generation. 3D model generation is a post image processing tool that develops models for groups of buildings (geoclusters) or a single building. There are different software that exist for 3D model development which vary in workflow, model production time and quality of output [8]. Current 3D modelling using building images is typically performed using LiDAR and RGB formats where IR images are overlaid onto the model.

A drone flight path will determine the image capturability through its distance from the test surface, bearing angle and altitude. These aspects are important to consider when developing a flight trajectory for the UAV system. Any adjacent buildings to the site under assessment may also hinder the image capturing process. Image processing is an important part of the drone analysis technique in order to create a comprehensive and complete image. Segmentation or photogrammetry is performed using either direct geo-referencing, ground control points, manual tie-in points or a combination thereof. The segmentation process can be affected based on-site elements and conditions (See Fig. 1).

In recent years, and especially with the technological development and price reduction in sensors and data acquisition systems, there has been a tremendous leap in all sorts of measurement activities within the built environment. The most common approach applicable in the built environment includes a group of algorithms known as Machine Learning, where data is analyzed through the construction of automated analytical models implemented in software.

More specifically, the types of ML algorithms used for anomaly detection in the building sector can be subdivided as supervised learning, and unsupervised learning. On top of the two main subdivisions of ML there are also more complex combinations of models including, ensemble methods, hybrid learning (Fig. 2).

Unsupervised Learning can be a number of approaches, but most notably, in this type of analysis only unlabeled data is used to make decisions on the data. Unsupervised techniques can include machine learning models that learn patterns and trends from unlabeled data, cluster analysis and other grouping methods that finds patterns and underlying features that describe groups to predict consumption observations. Several techniques of Unsupervised Learning include:

- Clustering: algorithms group a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups. Data in each cluster has similar statistical characteristics [14,15].
- One-Class Learning also known as unary classification or class-modelling, attempts to identify data points of a specific class among all data points, by primarily learning from a training set containing only the objects of that class. The task of fault detection is to detect whether the monitoring data belong to the normal class. Similarly, the task of fault diagnosis is to find which fault class the data belongs to [16].
- Dimensionality Reductions is the process of reducing the number of feature variables under consideration, by obtaining a (reduced) set of principal variables that describe the data optimally. Dimensionality reduction can be divided into feature selection and feature projection. Feature selection is usually based on ranking of features and is used to find a smaller subset the original set of variables which can be used to model problems more robustly. On the other hand, future projection methods are used to reduce the data in a high dimensional

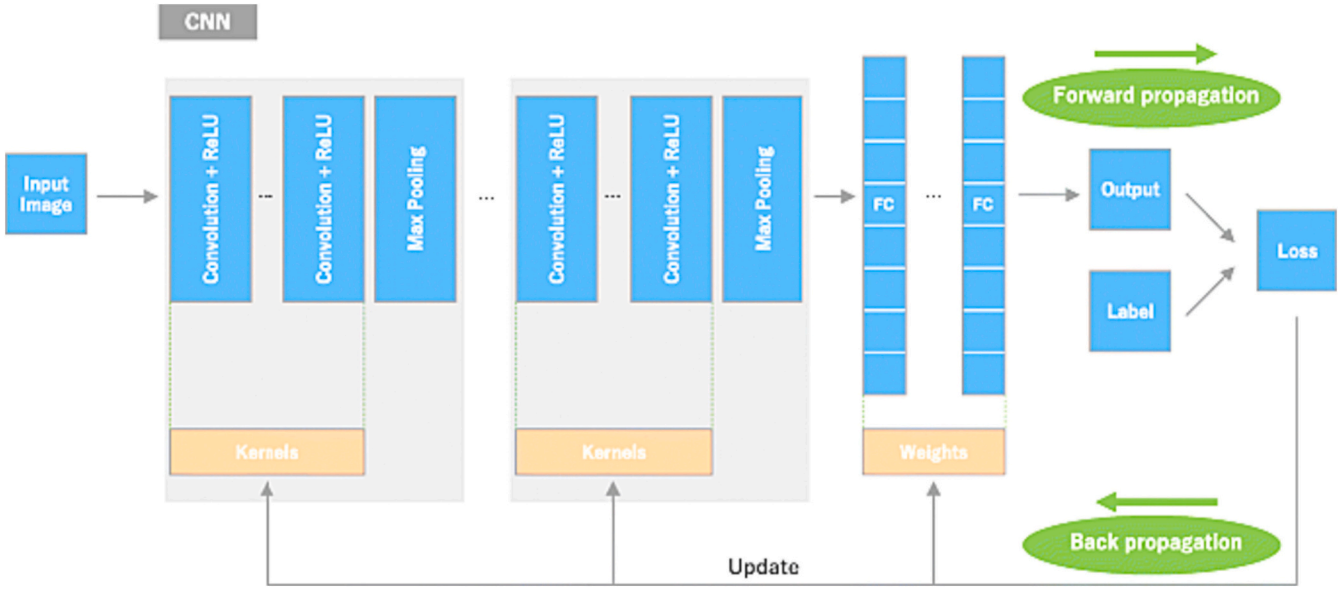


Fig. 3. Overview of a CNN architecture and training process. [9].

space to a lower dimension space usually through projection onto basis vectors-[17].

On the other hand, supervised learning is a machine learning paradigm for problems where the available data consists of labeled examples, meaning that each data point contains features and an associated label. Given a large sum of annotated data points, relationships can be learnt between input and output, features and targets respectively. Techniques of supervised learning include:

- Artificial Neural Networks (ANN) are an approach that attempts to mimic the capability of the human brain neurons. Each ANN consists of multi-layers (minimum two layers) of neurons and activation functions that form the connections between neurons [18]. Deep learning (DL) is the application of really large ANN networks, that are created by training with large datasets, and many model parameters. By including more layers of neurons, the number of parameters increases and lends to modelling more complex tasks. Over the last several years, convolutional neural networks have evolved and have been changing the landscape of computer vision. Such models solve for optimal filter (convolution) parameters, which are solved using gradient descent. The benefit of CNNs is that it incorporates neighbouring pixel information, which can lend to better image understanding.
- Traditional machine learning and classification refers to the development of a predictive model of previously developed classifiers such as Regression, Support Vector Machines and Random forest. As per supervised learning, this modelling problem where a class label is predicted for a given example of input data [19].
- Regression analysis is a tool for building statistical models that characterize relationships among a dependent variable. The machine learning method allows for the predicting of a continuous outcome variable [20].
- Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification that plots data in a n-dimensional space. The SVM method aims to determine a separating hyperplane that distinguishes positive examples from negative examples. Given a set of labeled training data, it generates input-output mapping functions which are used for classification [21].

In addition, Ensemble Methods are techniques that create multiple models which are then combined to produce improved results. Boosting, Bagging and Stacking are types of meta-algorithms used mainly for regression and classification by reducing bias and variance to boost the accuracy. Feature extraction is a machine learning method which selects and/or combines variables into features, effectively reducing the amount of data that must be processed, while still accurately and completely describing the original data set.

A recent study from North Dakota, USA, used a Mask R–CNN algorithm to detect various objects of a façade from IR images [22]. Karaaslan et al. [23] performed a study for crack detection and segmentation on concrete using a modified SegNet architecture CNN model that however IR was not used. Barahona et al. [24] used a measurement system mounted on a vehicle, that acquired geotagged optical and infrared images from the street-side of buildings. Their research performed a binary classification CNN model that trained on 2000 labeled IR images and identified anomalies with a precision score of around 89.2% and 75.6% recall on a test dataset of 1184 images. Finally, Mayer et al. [25] used a k-means clustering model on 40,000 buildings captured using maps street view and aerial view, coupled with other input metrics to rank energy efficiency of buildings.

A recent study by Thrampoulidis et al. [26] describes a machine learning-based surrogate model to approximate optimal building retrofit solutions. In 2019, [27], published a comprehensive review of papers published between 2004 and 2019 on the implementation of AI in the fault detection and diagnostics of building systems, all focusing on mechanical, electrical and control systems in buildings.

While the building sector has just recently begun to implement ML into the auditing field, other disciplines have much further advanced with the use of these analysis tools. The field of medicine for example, has been using IR thermography and ML applications for breast cancer detection, diabetic foot disease, skin cancer, carpal tunnel syndrome, dry eye disease, back pain, to name only some [28]. The main ML algorithms used for these applications include:

- Artificial Neural Networks are an approach that attempts to mimic the capability of the human brain neurons to process and handle large amounts of data so as to perform specific complex tasks. Each ANN consists of multi-layers (minimum two layers) of neurons and activation functions that form the connections between neurons [18].
- k-Nearest Neighbour (KNN) is a supervised, traditional classification, machine learning technique which has no explicit training phase. KNN's purpose is to use a database in which the data points are

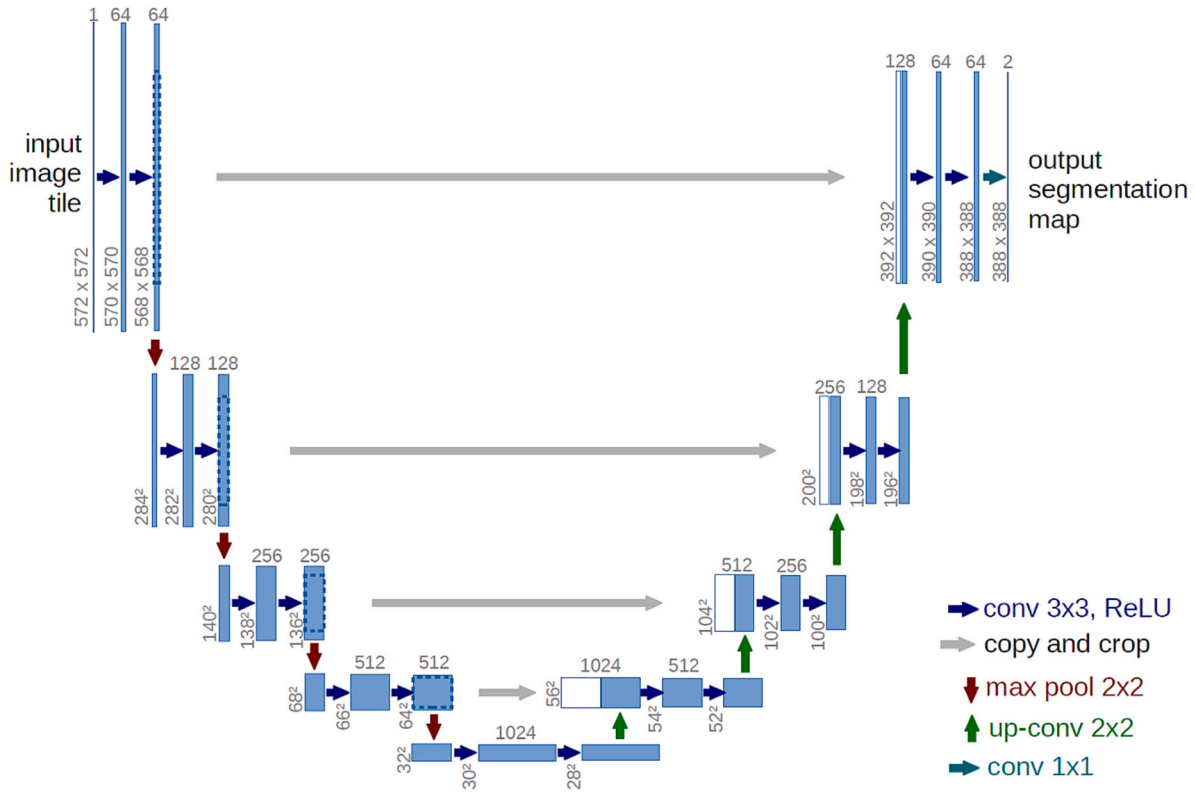


Fig. 4. U-NET Architecture [10].

separated into several classes to predict the classification of a new sample point. KNN algorithm is based on feature similarity. How closely out-of-sample features resemble the training set, determines how closely a given datapoint is classified.

- Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification that plots data in a n-dimensional space. The SVM method aims to determine a separating hyperplane that distinguishes positive examples from negative examples. Given a set of labeled training data, it generates input-output mapping functions which are used for classification [21].

2.1. Convolution neural networks

Deep Learning which has emerged as an effective tool for analyzing big data, uses complex algorithms and ANN's to train machines/computers, classify and recognize data similar to a human brain. In particular, a CNN is a type of ANN, which is widely used for image/object recognition and classification [29]. Deep Learning thus recognizes objects in an image by using a CNN.

CNN is a mathematical construct, inspired by the organization of animal visual cortex, [30,31] intended to automatically and adaptively learn spatial hierarchies of features, from low- to high-level patterns. CNNs are typically composed of three types of layers also known as building blocks: convolution, pooling, and fully connected layers (Fig. 3). Convolution and pooling layers perform feature extraction, whereas the fully connected layers map the extracted features into final outputs such as classification. A convolution layer plays a major role in CNN models, which is composed of a stack of mathematical operations. An example is a convolution which is a specialized type of linear operation. In digital images, pixel values are stored in a two-dimensional (2D) grid or mathematically, an array of numbers and a small grid of parameters called a kernel. An optimizable feature extractor is applied at each image position, which makes CNNs highly efficient for image processing, since a feature may occur anywhere in the image. As one layer feeds its output into the next layer, extracted features can hierarchically and progressively become more complex. The process of optimizing parameters such as kernels is called training, which is performed in order to minimize the difference between outputs and ground truth labels through an optimization algorithm called backpropagation and gradient descent.

The convolutional layer is a filter that passes over the input image, scanning a few pixels at a time and creating a feature map that is used to predict the class to which each feature belongs. The main task of the convolutional layer is to detect local conjunctions of features from the previous layer and map their appearance to a feature map. As a result of convolution in neuronal networks, the image is split into perceptrons, creating local receptive fields and finally compressing the perceptrons into feature maps., This map stores the information where the feature occurs in the image and how well it corresponds to the filter. Each filter is trained in regard to the position in the image it is applied to.

The Pooling layer also known as down sampling, reduces the amount of information in each feature obtained in the convolutional layer. The pooling layers are responsible for reducing the spatial size of the activation maps. In general, they are used after multiple stages of other layers like convolutional and non-linearity layers in order to reduce the computational requirements progressively through the network as well as minimizing the likelihood of overfitting. The key concept of the pooling layer is to provide translational invariance since particularly in image recognition tasks, the feature detection is more important compared to the feature's exact location.

The fully connected layer is usually the last building block in a CNN model and is comprised of several fully-connected neurons. The output feature maps of the layer before the fully connected layer are flattened, meaning they are transformed into a one-dimension (1D) array of numbers. The fully connected layer maps the extracted features from the convolution and pooling layers onto the final output. The hyperparameters and architecture of the building blocks can all be

Fig. 5. UAV & Camera: Falcom 8+ UAV (left), DJI ZENMUSE H20T Camera (right) (Product Brief: Intel® FalconTM 8.

manipulated and adjusted to improve performance of a model however, these three aforementioned steps are the building blocks of CNN models.

Region-Based Convolutional Neural Network or R-CNN is a type of machine learning model that is used for computer vision tasks, specifically for object detection [32]. This approach utilizes bounding boxes across the object regions, which then evaluates convolutional networks independently on all the Regions of Interest (ROI) to classify multiple image regions into proposed classes.

Mask R-CNN is another CNN developed on top of Faster R-CNN and is a state-of-the-art in terms of image segmentation and instance segmentation. The computer vision task Image Segmentation, is the process of partitioning a digital image into multiple segments or sets of pixels. This segmentation is used to locate objects and boundaries. There are two main types of image segmentation that fall under Mask R-CNN which include Semantic Segmentation and Instance Segmentation. Semantic segmentation classifies each pixel into a fixed set of categories without differentiating object instances meaning it deals with the identification/classification of similar objects as a single class from the pixel level. Instance segmentation or instance recognition on the other hand deals with the correct detection of all objects in an image while also precisely segmenting each instance. Mask R-CNN models have shown exceptional computational speeds which allows for real time results of the two segmentation processes.

2.2. U-NET

The research presented in this paper falls under the semantic segmentation portion and uses a specific CNN model called U-NET to perform the operations. U-NET is an architecture that was designed for fast and precise image segmentation by Ronneberger et al. [10]. The reason that U-NET was selected for this paper is because the model has shown success with limited input data over other CNN algorithms. U-NET is a U-shaped encoder-decoder network architecture, which consists of four encoder blocks and four decoder blocks that are connected via a bridge (Fig. 4).

The encoder network also called the contracting path, cuts the spatial dimensions by two and doubles the number of filters, or feature channels, at each encoder block. Similarly, the decoder network doubles the spatial dimensions and half's the number of feature channels. The architecture's U-shaped visual representation is where its name come originates from. The encoder network acts as the feature extractor of the model and learns an abstract representation of the input image through a sequence of the encoder blocks.

Each encoder block consists of two 3×3 convolutions, where each convolution is followed by a ReLU (Rectified Linear Unit) activation function. The ReLU activation function adds non-linearity into the system, which aids in the enhanced generalization of the training data. The output of the ReLU acts as a skip connection for the corresponding decoder block. Next, follows a 2×2 max-pooling, where the spatial dimensions, height and width, of the feature maps are reduced by half. This reduces the computational requirements by decreasing the number of trainable parameters. Skip connections provide additional information that helps the decoder to generate better semantic features. They also act as a shortcut connection that helps the indirect flow of gradients to the earlier layers without any degradation. Simply stated, the skip connection helps in improving the flow of gradient while back-propagation helps the network learn better representation. The bridge connects the encoder and the decoder network and completes the flow of information. It consists of two 3×3 convolutions, where each convolution is followed by a ReLU activation function. The decoder network is used to take the abstract representation and generate a semantic segmentation mask.

While a data set of images are required as input into a U-NET model, ground truth, masks are also required. These are 2 dimensional images of the same dimensions as the input data that accompany each image in the training set of the model. These masked images are binary and manually created where the 1's show the areas of interest and the 0's show all the remaining background pixels. Similar to the masks, the outputs of a U-NET model are predicted binary segmented masks that show areas of interest to unobserved data. The accuracy and results of the U-NET model are then observed and evaluated through the testing set for optimal performance and enhancements.

3. Methodology

3.1. Image acquisition

The data set selected for the U-NET application process performed in this research was acquired through in-kind contributions with an existing repository of jpg. File type IR images. Because the images were originally acquired for another purpose by an industry partner, the research team had not had any input in the process which was later addressed in the limitations. For the purpose of this study, the team selected images according to the following criteria from the much larger database: opaque façade systems (no glazed buildings); all vertical facades; some images with no façade deficiencies detected and all remaining images with defects; limited noise/interferences; captured using same camera for consistent colour palette.

The images were obtained with limited wind (below 0.5 m/s at 10 m height), which was needed for the UAV operation. Lateral radiation patterns and re-reflections of energy losses were avoided by selecting isolated buildings, limiting the inter-building radiation effect.

The images had a 512×640 resolution, where each pixel represents three values, RGB, ranging from 0 to 255. These images were attained using an Intel Falcom 8+ UAV coupled with a DJI ZENMUSE H20T camera (Fig. 5). The radiometric thermal camera outputs images set to an Ironbow colour pallet. Ironbow uses colours that run from black through blue for cooler temperatures to magenta, orange, yellow to bright white for warmer temperatures. The buildings were all assessed under regular operating conditions when the indoor and outdoor temperature differences were a minimum of 10° Celsius. A total of 142 images were selected from a data base of images, comprising of 14

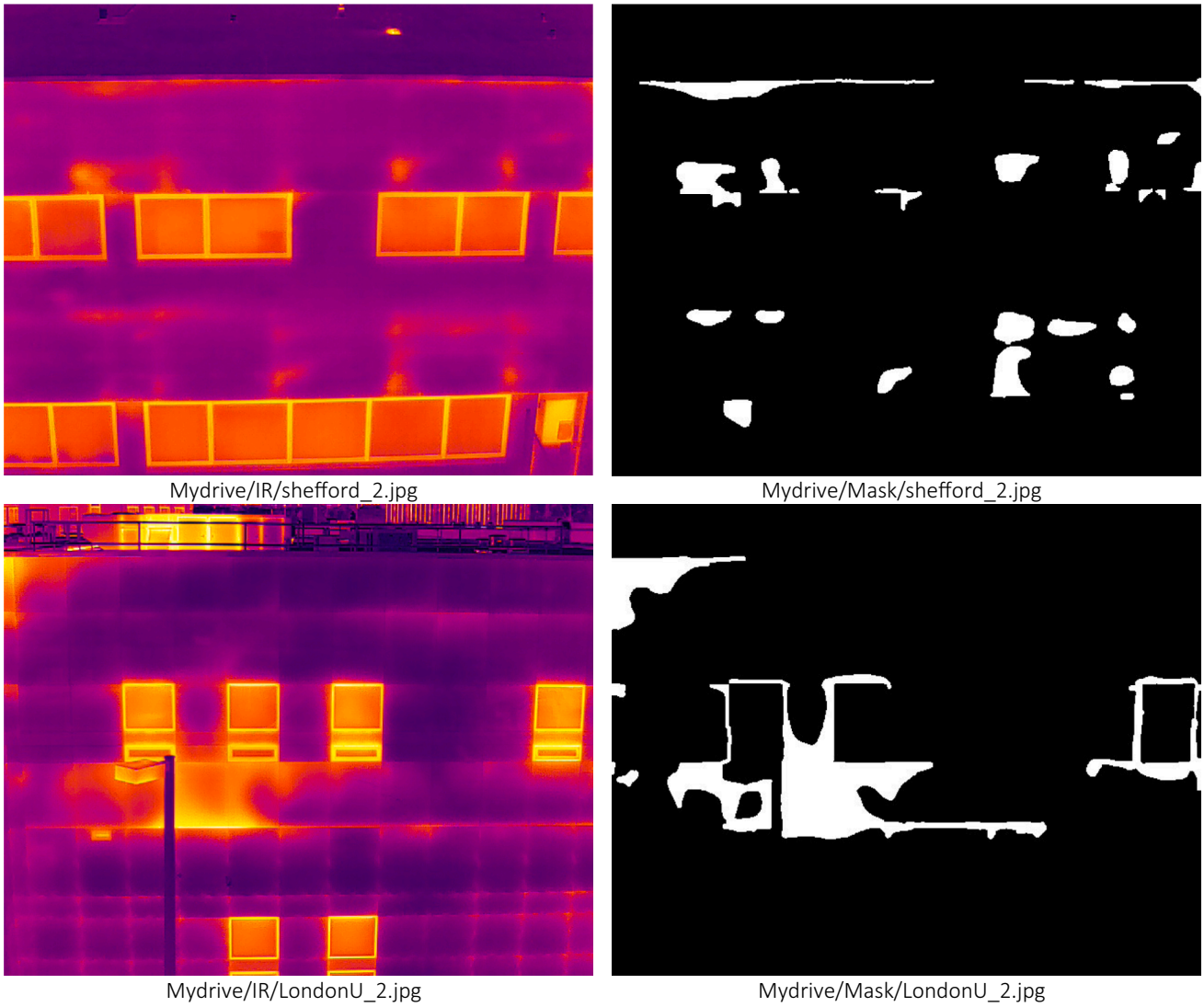


Fig. 6. IR and Mask Image: IR Image (left) with accompanying binary mask (right) examples and file directory/naming convention.

buildings having various images showing energy loss and additionally.

3.2. Data preparation

The data preparation process included creation of masks and segmentations, as it is a requirement for providing ground truth to the analytical model to learn from. The following two approaches were used: Temperature Threshold and Manually Created using Paint.exe [33].

The approach that was used to create the binary masks for the input images was performed using an application called FIJI using the Labkit plugin [34]. The user interface allowed the manual drawing process for each image to be much faster than Paint.exe. Using the FIJI software, each IR image was imported and a mask, highlighting areas of concern, were created. Locations where detailing and assembly design original to construction were considered poor, were avoided to focus on deteriorations and inconsistencies. Once the masks were imported into the model, it was observed that the images were not binary. Therefore, the following function was required to convert the masks into binary images where 127 (the halfway point between 0 and 255) was used as the split value: *def mask_to_binary(x):if x < 127: return 0. return 1.0.*

Once all the masks were created, the directory organization and file naming conventions were addressed in order to avoid errors. Fig. 6 presents examples of IR images and accompanying binary images as well as file naming conventions.

There are several elements and preprocessing procedures that must be performed in order to prepare the input data. Once all the IR images and accompanying masks are loaded into the model, they are made into their own lists and sorted so that both lists are matching with corresponding names. If a manually created directory of test set or validation set images has been made, this function is similarly applied and saved with their appropriate list names. These are performed using the Load-Data function as found in appendix A.

The IR images and binary masks are converted into an array of 1 channel where the binary image colour values remain 0 or 1 but the IR images are greyscaled. This is called dimensionality reduction. In this case RGB channels are collapsed into a single greyscale channel. Where RGB images are three channels and greyscale images are 1 channel, this function is performed to reduce computational requirements while preserving the information of the building components and surrounding elements in the image. On top of the channel reduction, the image size (Height x Width) values are adjusted as well to create a square image with aspect ratio 1:1. The 512 × 640 images are resized to either 128 × 128 or 256 × 256 or 512 × 512. For the purpose of this research, the image size was kept at 128 × 128. The greyscale images are then scaled down, meaning that the value of each pixel is divided by 255 to set the colour range between 0 and 1. Scaling image inputs is an important step which ensures that each input parameter (pixel, in this case) has a

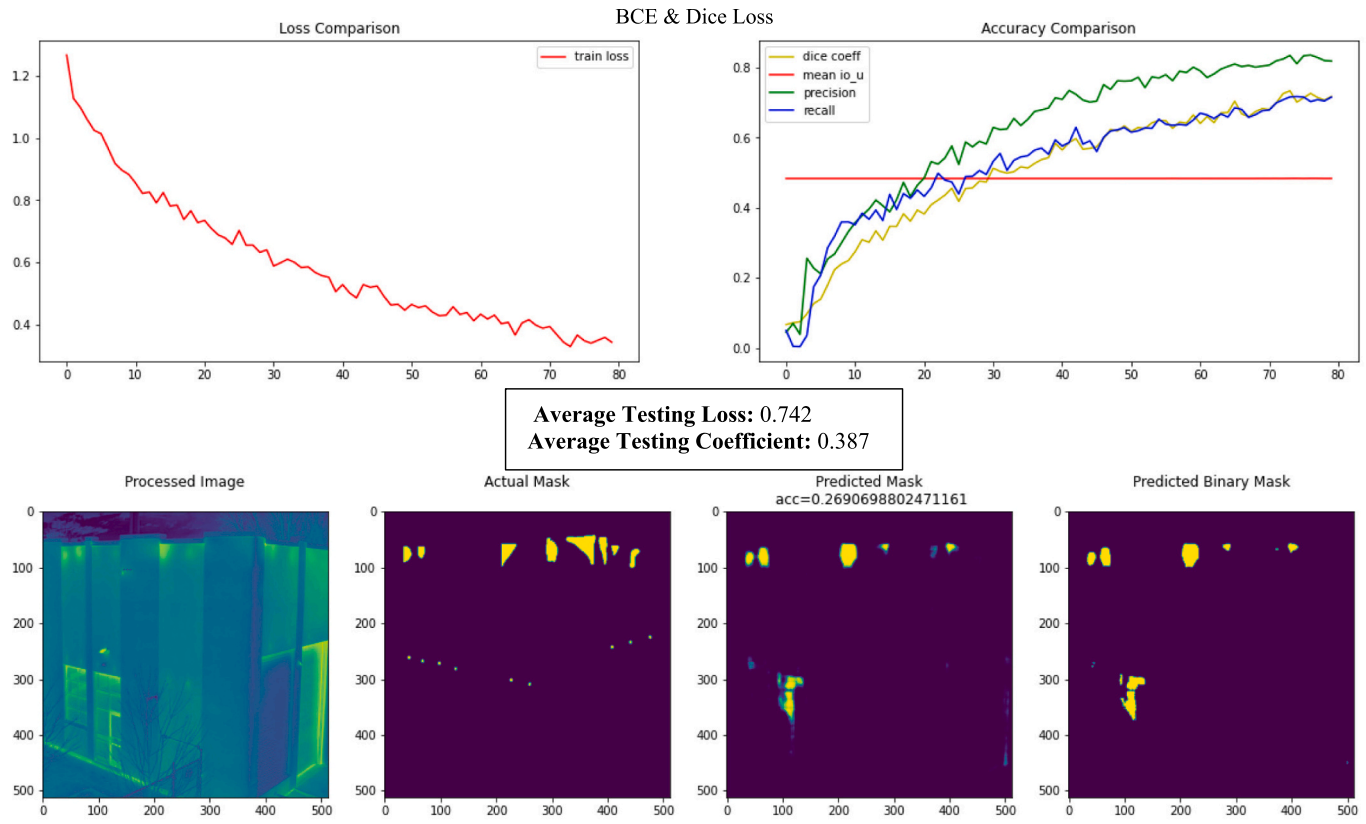


Fig. 7. Model 22 Results.

similar data distribution. This makes convergence faster while training the network. Data normalization is done by subtracting the mean from each pixel and then dividing the result by the standard deviation. The distribution of such data would resemble a Gaussian curve centered at zero.

Another data preprocessing step that can be applied is data augmentation. This technique involves augmenting the existing data-set with perturbed versions of the existing images. Scaling, rotations and other affine transformations are typical. This is done to expose the neural network to a wide variety of variations and can increase the training data by 2 or 4-fold. This makes it less likely that the neural network recognizes unwanted characteristics in the data-set.

3.3. U-NET architecture

The architecture used in this research was designed based off of Harshall Lamba's Salt Identification Case study and adjusted for application to this specific research [35]. The U-NET architecture is created using two functions: U-NET encoder mini block and U-NET model construct. These two functions are then used to build the U-NET model and are summarized to visualize all the layers and parameters.

First, the U-NET encoder mini block function is scripted. This is the "Conv 3x3 ReLu" operation depicted as the blue lines in fig. 6 and similarly defined as the "2 @ Conv Layers" in Fig. 7. This function applies 2 consecutive convolution layers with the following parameters that are defined and passed to it: Input tensor, number of filters, dropout probability, max pooling, kernel size and batch normalization.

The second function, U-NET model construct is then scripted. This function performs the remainder of the operations. This function ultimately defines the U-NET model and creates the contracting and expansion paths of the model with the following definitions for each notation:

- c1, c2, c3, c4, c5, c6, c7, c8, and c9 are the output tensors of the convolutional layers.
- p1, p2, p3 and p4 are the output tensors of max pooling layers.
- u6, u7, u8 and u9 are the output tensors of up-sampling also known as the transposed convolutional layers.

To summarize, the left-hand side is the contraction path or encoder where regular convolutions and max pooling layers are applied. In the Encoder, the size of the image gradually reduces while the depth gradually increases, starting from 128x128x3 to 8x8x256. This basically means that the network learns the "what" information in the image, however it has lost the "where" information. The right-hand side is the expansion path or, decoder, where transposed convolutions along with regular convolutions are applied. In the decoder, the size of the image gradually increases, and the depth gradually decreases, starting from 8x8x256 to 128x128x1. Intuitively, the Decoder recovers the "where" information (precise localization) by gradually applying up-sampling. To get better precise locations, at every step of the decoder, skip connections are used by concatenating the output of the transposed convolution layers with the feature maps from the encoder at the same level which in the code appear as: $u6 = u6 + c4$, $u7 = u7 + c3$, $u8 = u8 + c2$, $u9 = u9 + c1$. After every concatenation two consecutive regular convolutions are applied so that the model can learn to assemble a more precise output. On a high level, the following relationship is created: Input (128x128x1) $\geq$ Encoder $\geq$ (8x8x256) $\geq$ Decoder $\geq$ Output (128x128x1).

The output of the U-NET model construct function is the model itself, which is defined using the tf.keras.model function which is from the preset Tensorflow package. Keras is a DL application programming interface (API) written in Python, running on top of the ML platform TensorFlow [36]. While TensorFlow is an open-sourced end-to-end platform with a library for multiple ML tasks, Keras is a high-level neural network library that runs on top of TensorFlow. Both provide high level APIs used

```
unet.compile(optimizer="selected optimzer", loss="selected loss function", metrics=["selected metrics])
    if separate_valset == True:
     results = unet.fit("training IR images", "training Masks", batch_size="selected batch size", epochs="number of
       epochs", validation_data=("validation IR images", "validation Masks",))
    else:
     results = unet.fit("training IR images", "training Masks", batch_size="selected batch size", epochs="number of
       epochs",)
```

for easily building and training models. This function specifically, groups layers into an object with training and inference features. The Python3 scripted U-NET model construct function can be seen in appendix A as function name; UNetCompiled.

Finally, now that the two functions are created, the U-NET model is built. This U-NET build command calls the helper functions for defining the layers for the model, given the input image size. The U-NET model is assigned to a name and is written as follows:

unet = UNetCompiled(input_size=(resize_shape_x, resize_shape_y, 1), n_filters=32, n_classes=num_classes, batchnorm=True)

3.4. Training, validation & testing

Now that the U-NET model is ready, the training, validation and testing must be addressed. These are three sets of data that are used in ML practice. The training is the largest set of data, that is teaching the algorithm. It helps the model create and refine the rules using this data. It is a set of data samples used to fit the parameters of a machine learning model to training it by example. The model analyzes the dataset repeatedly to deeply understand its characteristics and adjust itself for better performance. The validation set is used in conjunction with training and helps understand how the model is performing throughout the training portion. The model does not learn from the validation set. Moreover, the validation is not required but is typically used to provide further insight for adjustments to hyper-parameters in the model and for tweaking the model structure. Finally, the testing set, is applied to the trained model and creates its own outputs to the new data based on how it has learned to perform.

In order to execute the model and apply the training and validation, the model must be compiled and then trained. Compiling the model requires taking the U-NET model and applying an optimizer, loss function and metrics. In order to train the model, the training data, batch size, number of epochs, and validation data (if applicable) must me be defined. These commands can be executed using the following Python3 code:

The .compile() function, as seen above, returns a specified source as a code object that is ready to be executed. In this case, U-NET is the model that is compiled with the specified parameters within the brackets. Optimizers update the model in response to the output of the loss function. Optimizers assist in minimizing the loss function. A loss function, also referred as cost function or error function, quantifies the error between the output of the algorithm and the given target value. In this case it will be the predicted masks and the manually created masks. Finally, metrics are used to monitor and measure the performance of a model during training, validation, and testing. Metric functions are similar to loss functions, except that the results from evaluating a metric are not used when training the model. Comparing training and validation metrics are important for understanding how the model is learning and performing.

There is a list of Keras optimizers that can be selected when creating a DL model. For the purpose of this research, ‘sgd’, ‘adam’, and ‘rmsprop’ optimizers were used during the simulations. These were selected based off of the example models. The hyperparameters and algorithms of these models are beyond the purpose of this research.

Several loss functions were used throughout this research endeavour. First, Keras' Sparse Categorical Cross-Entropy (SCC) and Mean Squared Error (MSE) were used. SCC computes the cross-entropy loss between the labels and predictions while MSE computes the mean of squares of errors between labels and predictions. Binary Cross Entropy (BCE), Dice Loss, and the two combined (BCE_Dice Loss) were also evaluated in this research. BCE computes the cross-entropy loss between true labels and predicted labels. Dice loss is 1 minus the dice coefficient where dice coefficient is two times (x2) the area of overlap divided by the total number of pixels in both images (predicted and ground truth mask). BCE_Dice is BCE plus dice loss. Finally, Tversky loss or generalized dice loss was used, which controls the contribution that each class makes to the loss by weighting classes by the inverse size of the expected region. A.

The unet.fit() function was used to train the model after it had been compiled. The batch size defines the number of samples in the batch. The batch is propagated through the network before updating the model parameters. After each batch, the model parameters are updated and a batch of samples go through one full forward and backward propagation.

3.5. Evaluating the model

The evaluation process is comprised of several elements. First, applying the trained model to the test set. This process provides new input data to the constructed model and using solely the IR images, predict binary masks are created. A model.evaluate() function is used to execute this process and once assigned to a variable the results can be viewed and evaluated. These predicted masks, similar to the validation metrics, as mentioned above, can be compared to the corresponding manually created masks to assess the accuracy of the model. Where validation and test set differ, is that validation is performed after each epoch, while testing is executed once the model has finished training.

4. Model development

This section presents an overview of the key developments made throughout this study. Each model discusses the hyperparameters that were used for model creation and clarify the reasons for the changes made.

Models 1–4. The first four models created were a trial-and-error approach to compiling some sort of successful results in the predicted masked images. The optimizers, loss functions and metrics were manipulated along with the batch size and number of epochs.

Models 5–13. In this batch of models, the following parameters were adjusted: new loss functions were added, the batch size and epoch sizes were manipulated. The data sets were also changed from a randomized set to a manually created training and testing set. However, Models 5 through 13 showed similar bias results as model 4. For that reason, these outcomes can also be disregarded. It was only after model 14 that the data sets were adjusted to avoid any overlap between training and testing sets. Mirroring was also added as a data augmentation function to the training set. Similar to the previous models, the graphs and predicted results all appeared to have strong results however, these can be ignored due to bias from overlap.

Models 14–22. It was after model 13 that the bias caused from overlap between training and testing sets was addressed. Therefore, the

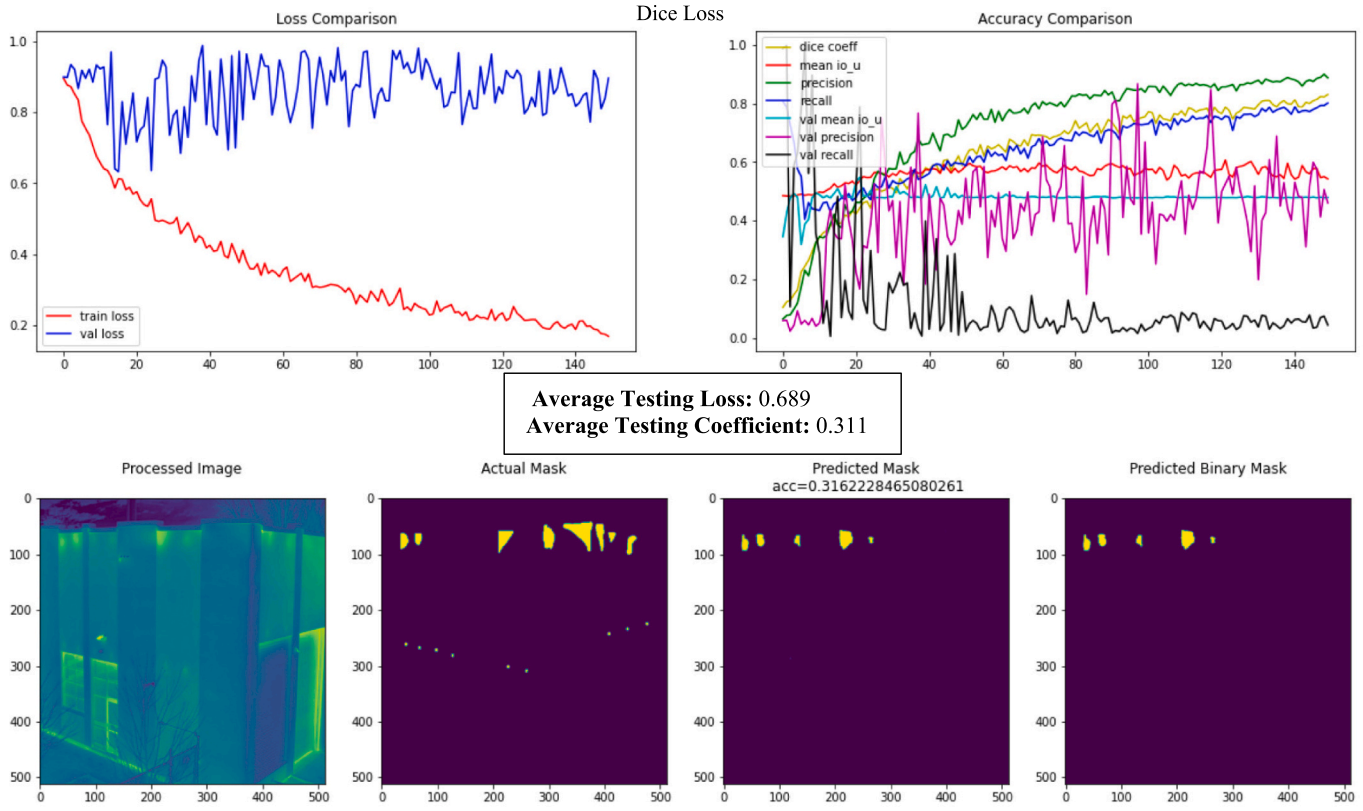


Fig. 8. Model 24 Results.

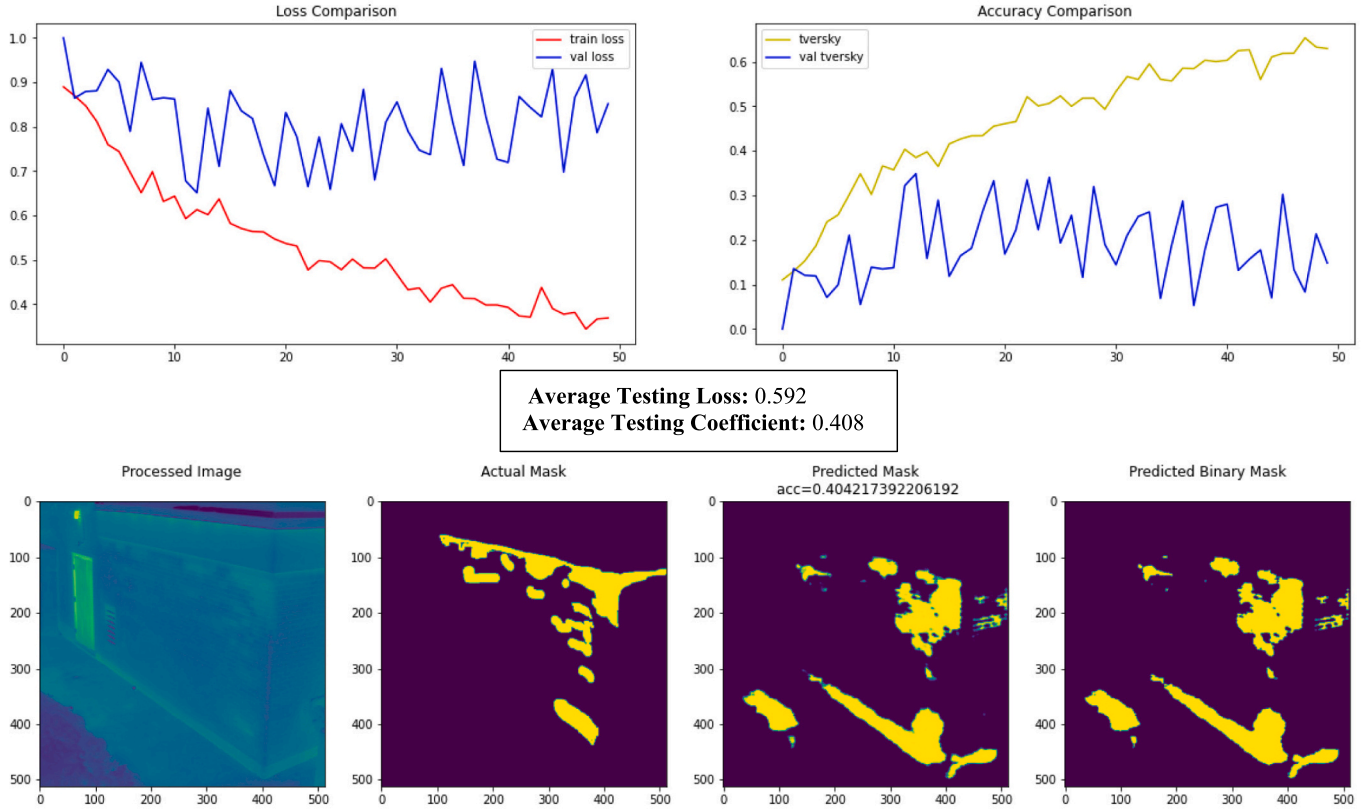


Fig. 9. Model 31 Results.

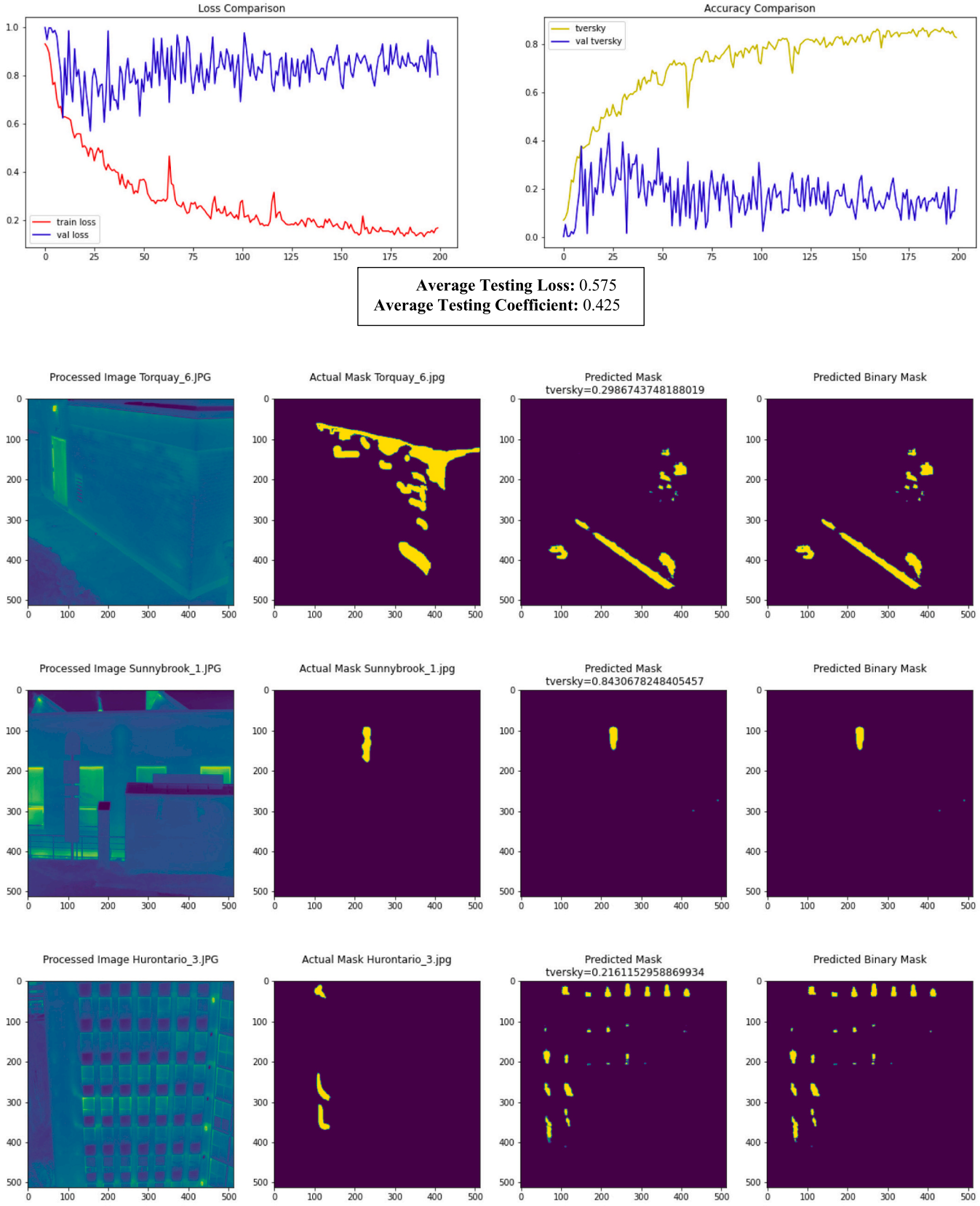


Fig. 10. – Final Model 36 Results.

28 selected images were reselected. This time, it was ensured that there were no building elements that appeared in both the training set and testing set. This avoided any sort of cheating and bias of the testing results. Once the overlap between data sets was removed, it was noted that the results and accuracy of the model was significantly worse. Models 14, 15, and 16, were using an input image size of 128 × 128. An adjustment was made to increase the input sizes in order for the model to retain more information and data from each image. 256 × 256 and 512 × 512 input image sizes were tried. The results improved slightly as the input image sizes increased. The following model parameters were used for model 22 which was noted to have the best results out of the nine as seen below:

Optimizer: Kera's Adam **Batch Size**: 5

Loss Function: BCE_dice_loss **Epochs**: 80

Metrcs: Dice_coefficient, Mean IOU, Precision, Recall **Training**: 228

Testing: 28 **Validation**: N/A

Other: Mirroring applied, 512 × 512 input size.

Models 23–25. While the current model was showing results of roughly 38% based on the average testing coefficient, it was important to understand more about the training process of the model. For that reason, a validation set of images was created to review the model's training process. Ten images were taken from the training set and put into its own directory for validation. This reduced the number of training images by 20, due to mirroring. The validation set when applied to the same three loss functions, BCE, Dice, and BCE_Dice combined, showed serious issues of overfitting. The parameters and results of these cases are as follows:

Optimizer: Kera's Adam **Batch Size**: 5

Loss Function: Dice_Loss **Epochs**: 150

Metrcs: Dice_coefficient, Mean IOU, Precision, Recall **Training**: 208

Testing: 28 **Validation**: 10

Other: Mirroring applied, 512 × 512 input size.

Models Cats & 26. In order to understand whether the poor accuracy of the model is due to the model hyperparameters or the data, a new dataset was presented. The Oxford Pets Dataset with annotations as used by Vidushi Bhatia was used for a comparison. Similar to the IR images, three directories were created with a matching number of images in each. The results of the Oxfords Pets Dataset' Cats images comparison demonstrated that the model functions well and presents high accuracy results when applied to a new, simpler, data set. To address this, the alternative model was created and executed on both the IR image dataset and the Cats data set for comparison.

Models 27–29. Coming back to the original model, another method that was explored is patching as described in section 3.2.3 Preprocessing. The patching data augmentation steps were executed using the original U-NET model with the same three loss functions. It was noted that with patching being used for data augmentation, the model was producing blank predicted segmentations for the majority of the results. This illustrates that the patching was confusing the model and making it more challenging to learn and understand the ground truth.

Models 30–31. The next approach that was made, was applying Tversky loss as the loss function. However, it was further noted that when α and β values are equal to 0.5 in eq. 2 below, it simplifies into the dice coefficient. For that reason, models 30, 31, and 32 were continuing to use dice loss as the loss functions. These three models are actually the same as model 24 above except for the number of epochs. It was noted that increasing the number of epochs resulted in less predictions when evaluating the 28 testing images. Since increasing the epochs resulted in increased FN, generalized dice loss or Tversky loss, was then actually introduced. This was done in order to balance the ratio of false negatives and false positive for more consistent and accurate results. The parameters of model 32 and results are as follows:

Optimizer: Kera's Adam **Batch Size**: 5

Loss Function: Dice_Loss **Epochs**: 50

Metrcs: Dice_coefficient, Mean IOU, Precision, Recall **Training**: 208

Testing: 28 **Validation**: 10

Other: Mirroring applied, 512 × 512 input size.

Models 32–35. These four models were the trial-and-error process of finding the perfect balance between Tversky loss values and the number of epochs. Different approaches were used. Since it was noted previously that increasing the number of epochs results in less segmentations, the number of epochs was increased. The results were presenting higher accuracy when the false negatives were increased as well, meaning that beta was increased, and alpha was decreased in the following equation:

$$TI = \frac{TP}{TP + \beta FN + \alpha FP} \tag{2}$$

Models 36. This model is the final and most accurate model developed during this research process. In this model, the alpha value and beta value were increased to 0.1 and 0.9 respectively. This was done to increase the sensitivity of the model and present a finer level of control for the model to learn. Therefore, the loss function used for this model was the Tversky loss alpha = 0.1 and Beta = 0.9. The optimizer used for this model is the Kera's Adam optimizer. The metrics used to evaluate the model is the Tversky coefficient which is 1 minus the Tversky loss. The batch size selected for the model is 5 and the number of epochs used is 200. The training dataset is 104 images but then doubled using the vertical mirroring augmentation function. The validation set used was 10 images and the testing set used was 28 images. The input image size used for this model was 512 × 512. The results of the final model are as follows.

5. Discussion

As previously mentioned, each model, when tested, developed a predicted mask for every image in the testing dataset. For clarity, only selected images were presented in order to better understand the accuracy and development and therefore, some assumptions were made based off all predicted images collectively. The following section discusses the pros and cons of each model and how the specific characteristics affected the development process. The final paragraph of this section presents the key take always from this research process while each model discusses the learning process through development.

Model 1 – The first model developed presented blank predicted segmentations on the test set and therefore, did not work. The graphs also did not show any sort of training from the model. This was due to the hyperparameters requiring adjustments as well as the 3rd channel displaying the borders which was causing issues.

Model 2 – Using temperature threshold as the input region segmentation on the same model proved that the U-NET model was in fact working, but just needed to be tweaked. The Graphs in this simulation showed that the model was managing to learn however predicted results showed challenges. This was determined to be due to the loss function being used, Sparse Categorical Cross Entropy is best used for distribution-based models whereas this is research is a region-based model.

Model 3 – For model 3, the loss function was updated to a Binary Cross Entropy and Dice Loss combined function. This showed a lot of improvement with the predicted segmentation results. The images were showing some sort of consistency but were very inaccurate. The training loss was also noted to be trending in the right direction, meaning the model was beginning to learn.

Model 4 – This model, the graphs were looking strong when the batch size was reduced to 5 and the epochs were increased to 80. The predicted results were also accurate however this was due to the randomization and bias results. The accuracy of this model is irrelevant due to overlap of images.

Model 5–7 – Binary Cross Entropy and Dice Loss were added as loss functions for this model. Increasing the epochs showed similar results,

however due to randomization of the training and testing sets plus the overlap, the results between models are not accurately comparable. These models can also be considered irrelevant due to overlap of images.

Model 8–10 – Mirroring as a method of data augmentation was added in these models. Mirroring the training data set along the vertical access showed some minor improvements and was not making the model worse, therefore, the function was kept for the remainder of the models when applicable. These datasets are again randomized every time and are therefore not accurate for comparison. These models can also be considered irrelevant due to overlap of images.

Model 11–13 – In these models, the randomization was removed, however, overlap was still noted after viewing the results meaning that bias of the results is still present. Due to the overlap between training and testing images the graphs and predicted masks are accurate and are to be considered irrelevant.

Model 14–16 – For these models, the overlap between training and testing sets was removed, meaning all results are unbiased and valid for the remained of the models. Average testing loss and testing coefficient values were also implemented for added accuracy values. The loss needs to be low (close to 0) and coefficient needs to be high (close to 1), since coefficient is 1 – loss. Coefficient can be considered the accuracy and should therefore, ideally be close to 1 (100% accuracy). The graphs from the models were decent however without validation curves it is difficult to tell how the model will predict unseen data. For that reason, looking at the average testing loss and coefficient values, it is clear that they are very inaccurate.

Model 17–19 – For these three models, the input image data sizes were increased from 128 × 128 to 256 × 256. The average losses and coefficients were noted to be better with 128 × 128, however that is most likely due to the pixel sizes losing information, but when looking at the predicted images, they show slightly more consistent ground truth results.

Model 20–22 – For these three models, the input image data sizes were increased from 256 × 256 to 512 × 512. The computational time increases significantly, but the model appears to be learning the ground truth better and more consistently, even though the values say testing values indicate otherwise.

Model 23–25 – Since it was challenging to understand how the model was functioning based on the standing assessment methods, a validation set was added. While the loss comparison graphs for BCE and BCE & Dice Loss appear to be skewed, the reason for such a large initial drop is due to the first epoch initialization being very high. The validation trends are showing overfitting and are not trending in the same fashion as the training loss. Overfitting is noted when the validation loss remains high and presents a large gap between the training and validation. Overfitting occurs when a model fits too well to the training set. It then becomes difficult for the model to generalize to new examples that were not in the training set. This means that the model recognizes specific images in the training set instead of learning general patterns.

Model 26 – The alternative U-NET model was used for this case. The results were similar to the original U-NET model therefore, the focus remained with the original model.

Model 27–29 – Patching was applied to these models as a data augmentation strategy. The results noted were poor and the predicted masks were all blank. This strategy would require further exploration however the results from the patching test cases that were done did not show any sign of success.

Model 30 – This model was an accidental rerun of the dice loss function; however, 200 epochs were used where are 150 were used in model 24. It was noted that the average loss and average coefficients of the testing sets were better with less epochs.

Model 31–50 epochs were used for this model run with the same dice loss function. The average testing loss and coefficient values were noted to be better however the model was over predicting when reviewing the predicted segmentations. Therefore, increasing the epochs resulted in less predictions however that is incorrect and there for a false negative. The model is generalizing better with more epochs which is good but underpredicting which is bad. Theoretically with a large data set, and introduction of new buildings, the segmentations would be better with higher epochs. However, since no increase in data is possible, the Tversky loss must be used. To account for the increased false negatives, alpha and beta are to be changed to help balance the loss function equation. Looking at eq. 2, beta account for the percentage of false negatives, and alpha for the percentage of false positives. For that reason, the beta and alpha values are adjusted to create a more generalized dice loss also known as Tversky loss.

Model 32 – Tversky loss was applied in this model. Alpha was set to 0.3 and Beta was set to 0.7. The epochs were kept at 50 to observe the results. It was noted that with dice loss generalization, the model was managing to learn better and provide better accuracy for the testing set. The validation graphs continued to show overfitting.

Model 33 – When the epochs were increased to 200 for this model, the model was noted to predict less, more false negatives. However, with the Tversky loss, the false negatives are penalized which together balances out to a more accurate model with better segmentation results.

Model 34 – When alpha and beta were changes to 0.4 and 0.6 respectively, the average loss and coefficients of the testing set were noted to be worse. This means that the generalization was accounting in the wrong direction and needed to be adjusted.

Model 35 – When alpha was adjusted to 0.2 and beta to 0.8, the averages were noted to be better for the testing set. While the graphs are still showing significant signs of overfitting, the predicted results were noted to be as a majority better.

Model 36 —In this case with the same hyperparameters as before, the generalization of the Tversky loss was again adjusted to alpha being 0.1 and beta 0.9. The average testing loss and coefficient values were noted to be the best values achieved from this entire research, providing a roughly 42.5 accuracy. The predicted testing segmentations were observed to be excellent in some cases and very poor in others. This is most likely due to the overfitting as seen in the graphs. Overfitting mainly due to a dataset issue. This was also similarly noted when the Cats comparisons were used and showed much greater results with 20,000 images compared to the 144. Further fine tuning could have been performed to find even more optimal alpha and beta values, however computational resources were a limitation and without an updated dataset.

This model development process was determined to very successful and promising as a novel study and shows a lot of promise for future work. First and foremost, the results of the later models presented that the training process was managing to find patterns and learn to a certain extent. It was the challenging ground truth/noise accompanied by the limited data that was the main setback for the accuracy. The ground truth was determined to be too general. A more defined and process ground truth needs to be developed for the model to be able to learn and properly predict. For example, CATs, is a very distinguishable region to segment and differs so much from other items on a 2D image. There is much less noise in a Cats image that will trouble a CNN model to learn and predict accurately. For that reason, the ground truth and its level of difficulty must be reduced for future work.

First and foremost, the dataset that was used and available for this research was collected from an existing repository made accessible to this research team through in-kind contributions. Noise causes more pattern difficulties and affects the abilities of the model to learn. Less noise will result in easier learnability and ultimately better results. This criterion greatly limited the amount of IR images deemed suitable for this research. On top of that, although the selected facades appeared similar and of the similar age and style of construction, it was impossible to verify the assembly design and construction type, which could, vary significantly. A more appropriate approach to this application, to reduce variables, would be to collect datasets and build models based on architypes. This would allow for greater constancy between images.

Another important factor that was not considered because the

information was not available, is the data on pressurizing of the buildings during assessments. The interior pressure of a building will drastically affect the thermal results captured by an IR camera.

The Cats dataset for example, as seen in two of the models presented in the Results chapter, demonstrate excellent accuracy when applied to the same models created for the purpose of IR. It was also noted that having a significantly larger dataset of Cats for the model to learn from, that the accuracy became even better. This establishes that the limited dataset of IR images caused a serious issue for model training in this research. The main reason that the models developed in this research were not accurate were most probably due to the challenging ground truth. This must be consistent for each instance, which would be simpler to execute with similar building architypes. Defining the ground truth also translates to defining the noise.

Another element to consider with regards to image quality, is the resolution and scope of view. The dataset that was used, provided images of very close buildings and others where most of the building was in the field of view. Prior to capturing the images, it is important to set a flight path and define the image capturing distance from the building under assessment. When evaluating the image capturing processes of other industries like biomedical engineering, the CNN models are trained using body imaging with specifically the same fields of view. This demonstrates the importance of flight path and angle of assessment, when capturing building envelope conditions.

The following items were determined as limitations that affected the results of this research.

- The available IR images that matched the scope of this investigation were limited due to the lack of involvement in the acquisition process and set criteria;
- Resolution and scope of view of the IR images was a limitation and inconsistency for this study;
- Verifying the assembly design and construction type was impossible, which, varies significantly from one building to another;
- U-NET was the only CNN model that was investigated;
- U-NET in-depth adjustments to architecture and hyperparameters were not performed due to a limitation in computer science background;
- Computational power was a minor limitation during this study which limited the number of simulations that can be performed, quickly.

6. Conclusions

For years, IR imaging has been used as a non-destructive building envelope testing method. Coupled with UAVs plus advancements in drone and image capturing technologies, has boosted the building enclosure IR image capturing industry. With image procurement strategies increasing, the assessment process has remained unchanged. This means that all images are manually reviewed by building science experts, which is time consuming and inefficient. The importance of evaluating and rehabilitant existing high energy consuming buildings is imperative for the worldwide fight against climate change. For that reason, a more strategic and effective method for identifying and classifying building envelope deficiencies is imperative. With AI and ML becoming an ever so evolving and powerful means in the world of technology, there are extensive tools that can be applied to this IR assessment strategy.

The model development process demonstrated the potential that this DL strategy has on defect detection. With such a limited dataset used in this research, the trained model managed to, in some cases, very accurately segment areas of interest. While there were many testing predicted segmentation images that were inaccurate, the model still demonstrated a lot of potential. Adjustments of the hyperparameters presented an increase in accuracy demonstrating that the model can learn and increase in precision. For that reason, yes CNN DL analytical models can be applied to the review process of IR exterior building envelope images, however, significant further work is required.

With a data set of 144 IR images, the trained model managed to acquire an average testing loss of 0.575 and average testing coefficient of 0.425. The testing coefficient translates to roughly a 43% accuracy rate for region segmentation when compared to the manually created masks. This high average for initial research is strong and shows the potential that DL has to offer for this assessment strategy. These values were acquired with the Tversky loss function being used where alpha = 0.1 and Beta = 0.9. The optimizer used for this final model is the Kera's Adam optimizer. The metrics used to evaluate the model is the Tversky coefficient which is 1 minus the Tversky loss. The batch size selected for the model is 5 and the number of epochs used is 200. The training dataset is 104 images but then doubled using the vertical mirroring augmentation function. The validation set used was 10 images and the testing set used was 28 images. The input image size used for this model was 512 × 512. These hyperparameters used with the original U-NET model presented the best results (See Figs. 8-10).

The main contributor to the inaccuracy of the model was determined to be the dataset and ground truth consistency. Collecting and developing a more established dataset of IR building envelope images could drastically improve the accuracy and performance of the DL model developed in this research and as found in the appendix of this paper. With extensive future research opportunities, the future of this effective and non-intrusive building envelope assessment strategy is promising and will be a great contributor for global reduction of energy demands.

CRediT authorship contribution statement

David Gertsvolf: Resources, Investigation, Data curation. **Miljana Horvat:** Methodology, Project administration, Validation, Supervision. **Danesh Aslam:** Methodology, Validation. **April Khademi:** Software, Supervision. **Umberto Berardi:** Methodology, Project administration, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

The authors would like to thank QEAtech for the use of IR images. In addition, the authors acknowledge that partial research funding was obtained through the MITACS Accelerate program.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.apenergy.2024.122696.

References

[1] UNEP. Building design and construction: Forging resource efficiency and sustainable development. 2012.
[2] Berardi U. A cross-country comparison of the building energy consumptions and their trends. Resourc Conserv Recycl 2019;123:230–41. https://doi.org/10.1016/j.resconrec.2016.03.014.
[3] Berardi U. The development of a monolithic aerogel glazed window for an energy retrofitting project. Appl Energy 2015;154:603–15. https://doi.org/10.1016/j.apenergy.2015.05.059.
[4] Fox M, Coley D, Goodhew S, de Wilde P. Thermography methodologies for detecting energy related building defects. Renew Sustain Energy Rev 2014;40:296–310. https://doi.org/10.1016/j.rser.2014.07.188.

[5] Lucchi E. Applications of the infrared thermography in the energy audit of buildings: a review. Renew Sustain Energy Rev 2018;82:3077–90. https://doi.org/10.1016/j.rser.2017.10.031.
[6] Youcef MHAL, Feuillet V, Ibos L, Candau Y. In situ quantitative diagnosis of insulated building walls using passive infrared thermography. Quant InfraRed Thermogr J 2020;0(0):1–29. https://doi.org/10.1080/17686733.2020.1805939.
[7] Colomina I, Molina P. Unmanned aerial systems for photogrammetry and remote sensing: a review. ISPRS J Photogramm Remote Sens 2014;92:79–97. https://doi.org/10.1016/j.isprsjprs.2014.02.013.
[8] Rakha T, Gorodetsky A. Review of unmanned aerial system (UAS) applications in the built environment: towards automated building inspection procedures using drones. Autom Construct 2018;93:252–64. https://doi.org/10.1016/j.autcon.2018.05.002.
[9] Yamashita R, Nishio M, Do RKG, Togashi K. Convolutional neural networks: an overview and application in radiology. Insights Imaging 2018;9(4). https://doi.org/10.1007/s13244-018-0639-9. Article 4.
[10] Ronneberger O, Fischer P, Brox T. *U-Net: Convolutional Networks for Biomedical Image Segmentation* (arXiv:1505.04597). arXiv. http://arxiv.org/abs/1505.04597. 2015.
[11] Stoker J. Lidar point cloud. 2017. Washington, DC, https://www.usgs.gov/media/images/lidar-point-cloud-washington-dc-0.
[12] Fehmiu R. Seaports are seen using thermography technology with aerial drone cameras from above. Shutterstock.com.
[13] Maximilian B. Thermal scanning of new Multi-Story Building. Alamy Stock Photo.
[14] Capozzoli A, Lauro F, Khan I. Fault detection analysis using data mining techniques for a cluster of smart office buildings. Expert Syst Appl 2015;42(9):4324–38. https://doi.org/10.1016/j.eswa.2015.01.010.
[15] Du Z, Fan B, Jin X, Chi J. Fault detection and diagnosis for buildings and HVAC systems using combined neural networks and subtractive clustering analysis. Build Environ 2014;73:1–11. https://doi.org/10.1016/j.buildenv.2013.11.021.
[16] Yan K, Ji Z, Shen W. Online fault detection methods for chillers combining extended kalman filter and recursive one-class SVM. Neurocomputing 2017;228:205–12. https://doi.org/10.1016/j.neucom.2016.09.076.
[17] IEEE. A new approach to dimensionality reduction for anomaly detection in data traffic | IEEE J Magaz IEEE Xplore. Retrieved May 14, 2021, from, https://ieeexplore-ieee-org.ezproxy.lib.ryerson.ca/document/7580700.
[18] Khan SS, Madden MG. A survey of recent trends in one class classification. In: Coyle L, Freyne J, editors. Artificial Intelligence and Cognitive Science. Springer; 2010. p. 188–97. https://doi.org/10.1007/978-3-642-17080-5_21.
[19] Sial A, Singh A, Mahanti A. Detecting anomalous energy consumption using contextual analysis of smart meter data. Wireless Networks 2019:1–18. https://doi.org/10.1007/s11276-019-02074-8.
[20] Lee W-Y, House JM, Kyong N-H. Subsystem level fault diagnosis of a building's air-handling unit using general regression neural networks. Appl Energy 2004;77(2):153–70. https://doi.org/10.1016/S0306-2619(03)00107-7.
[21] Liang J, Du R. Model-based fault detection and diagnosis of HVAC systems using support vector machine method. Int J Refrigerat 2007;30(6):1104–14. https://doi.org/10.1016/j.ijrefrig.2006.12.012.
[22] Sadhukhan D, Peri S, Sugunaraj N, Biswas A, Selvaraj DF, Koiner K, et al. Estimating surface temperature from thermal imagery of buildings for accurate thermal transmittance (U-value): a machine learning perspective. J Build Eng 2020;32:101637. https://doi.org/10.1016/j.jobe.2020.101637.
[23] Karaaslan E, Bagci U, Catbas FN. Attention-guided analysis of infrastructure damage with semi-supervised deep learning. Autom Construct 2021;125:103634. https://doi.org/10.1016/j.autcon.2021.103634.
[24] Barahona B, Buck R, Okaya O, Schuetz P. Detection of thermal anomalies on building façades using infrared thermography and supervised learning. J Physics: Conf Ser 2021;2042(1):012013. https://doi.org/10.1088/1742-6596/2042/1/012013.
[25] Mayer K, Haas L, Huang T, Bernabé-Moreno J, Rajagopal R, Fischer M. Estimating building energy efficiency from street view imagery, aerial imagery, and land surface temperature data. Appl Energy 2023;333:120542. https://doi.org/10.1016/j.apenergy.2022.120542.
[26] Thrampoulidis E, Mavromatidis G, Lucchi A, Orehounig K. A machine learning-based surrogate model to approximate optimal building retrofit solutions. Appl Energy 2021;281:116024. https://doi.org/10.1016/j.apenergy.2020.116024.
[27] Zhao Y, Li T, Zhang X, Zhang C. Artificial intelligence-based fault detection and diagnosis methods for building energy systems: advantages, challenges and the future. Renew Sustain Energy Rev 2019;109:85–101. https://doi.org/10.1016/j.rser.2019.04.021.
[28] Vardasca R, Magalhaes C, Mendes J. Biomedical applications of infrared thermal imaging: current state of machine learning classification. Proceedings 2019;27(1):46. https://doi.org/10.3390/proceedings2019027046.
[29] Choudhary P, Pathak P. A review of convolution neural network used in various applications. In: 2021 5th international conference on information systems and computer networks (ISCON); 2021. p. 1–5. https://doi.org/10.1109/ISCON52037.2021.9702315.
[30] Fukushima K. Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol Cybern 1980;36(4):193–202. https://doi.org/10.1007/BF00344251.
[31] Hubel DH, Wiesel TN. Receptive fields and functional architecture of monkey striate cortex—Hubel—1968. J Physiol Wiley Online Library 2023. https://doi.org/10.1113/jphysiol.1968.sp008455. Retrieved September 29, 2022, from.
[32] Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation (arXiv:1311.2524). arXiv 2014. http://arxiv.org/abs/1311.2524.
[33] Gertsvolf D, Horvat M, Khademi A, Aslam D, Berardi U. Image Processing for Future Machine Learning Algorithm Applications on Infrared Thermography of Building Envelope Systems. 2022. p. 549–56. https://doi.org/10.1007/978-981-19-9822-5_58.
[34] Schindelin J, Arganda-Carreras I, Frise E, Kaynig V, Longair M, Pietzsch T, et al. Fiji: An open-source platform for biological-image analysis. 2019.
[35] Lamba H. Understanding Semantic Segmentation with UNET. Medium. https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47; 2019, February 17.
[36] Chollet F. Keras documentation: image segmentation with a U-Net-like architecture. https://keras.io/examples/vision/oxford_pets_image_segmentation/; 2019.