Deep Learning Based Wind-Induced Building Damage Classification

Aaron John
Electrical and Computer Engineering
Western University
Student ID: 251190234
ajohn527@uwo.ca

Abiy Melaku
Civil and Environmental Engineering
Western University
Student ID: 250903647
amelaku@uwo.ca

Tsinuel Geleta
Civil and Environmental Engineering
Western University
Student ID: 250954669
tgeleta@uwo.ca

***Abstract*—Wind-induced damage to buildings is one of the primary sources of hazard-related loss. Processing post-damage survey imagery of affected areas is important to quantify the incurred loss and suggest future improvements. Deep learning is becoming a viable option for building damage classification through image processing. This study proposes CNN based deep neural network for classification of damage scales per individual building using publicly available data collected from four hurricanes in the US. The model uses a stack of convolutional layers with batch normalization for feature extraction and linear layers to classify the outputs. Multi-class cross entropy is used for loss function. Evaluation presented using F1-score and balanced accuracy. Our model attained F1-score of 0.741 and balanced accuracy of 63.9%. Comparison of this result to previous studies with the same dataset showed that the current results are reasonably good.**

***Index Terms*—Wind-induced Damage, Deep learning, Convolutional Neural Network**

I. Introduction

Extreme wind events, such as hurricanes and tornados, are among the top contributors to the loss of lives and livelihoods related to natural hazards [1]. The wind-induced damage on buildings and other structures in past wind events is assessed to estimate future loss value, improve design methods, and in future emergency planning. Damage assessment involves two major steps: data collection and data processing. The data collection is done by either ground survey (taking conventional photograph with notes walking through the affected area), high-altitude aerial imagery using small-planes or large drones, versatile drone imagery and footage by flying them at low altitudes, or other specialized technologies like LiDAR. Once the data has been collected, it is used to classify the damage level of buildings and their components according to established damage classification scales [2].

The post-damage data processing to classify the degree of damage in buildings is conventionally done manually through visual assessment of images and consulting notes. Although this method is generally robust, it is slow and cumbersome to apply. Some types of data such as high altitude aerial imagery and LiDAR measurements are challenging, if not impossible, to process manually. The fast-growing computer vision technology in recent days is proving to be a viable option to solve these problems and open up new ways to process damage survey data fast.

In recent days, a growing number of researchers are applying deep learning and other machine learning techniques to process post-disaster data. Some works [2]–[5] worked on processing aerial imagery taken directly above the affected area. Others [6], [7] have considered video of the affected area taken by drones or helicopters at a relatively low altitudes to identify and classify damaged buildings. Reference [8] developed a model to detect individual affected buildings from 360^o panaroma pictures taken on the ground. Processing video data recorded with drones and helicopters with machine learning techniques was applied by [6], [7]. The most common method applied to process image or video damage date is Convolutional Neural Network (CNN) [3], [5]–[8]. Most studies used transfer learning with networks trained to improve the performance of their models. For instance, [6] transferred RetinaNet and MobileNet to their building detection and mobile vision components of their model.

In this study, a database of labeled post-disaster aerial image database to train, validate, and evaluate a CNN damage classification model. The data preparation stage involves splitting the aerial image into individual buildings and resizing each to equal sizes. The CNN model is trained on approximately 20,000 individual buildings labeled with four different damage levels. The performance of the model is evaluated using F1 score and balanced accuracy. These performance measures are commonly used for multi-class classification and account for biased training data.

This document is organized as follows. First, a brief background on application of CNN for image classification is provided in §II. It is followed by §III with the description of the methodology including description of the dataset, the CNN model, training, evaluation, and tuning processes. The results of the current model with discussion of observations and comparison to similar previous studies is provided in §IV. Finally, summary and conclusion is presented in §V.

II. Background

Reference [9] describes convolutional neural network (CNN) as:

A course project as part of partial fulfillment of ECE 9603/9063B by Prof. Katarina Grolinger. Fall term, 2021. Western University, London ON, Canada

> Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

A typical CNN architecture includes convolution layers, pooling layers, fully connected layers, and activation functions. The convolution, activation, and pooling layers are termed as feature extraction layers that represent the relatively higher order input into a simpler output with the important features highlighted. The convolution acts as a smoothing operation whereby the local differences are filtered through a set of weights called kernels. These kernels are modified during training to emphasize on the most important features of the input. The kernel size and stride of moving through the input are typically regarded as hyperparameters that are tuned. This convolution operation is a linear operation that captures linear relationships. In order to handle non-linearity, the output of the convolution is passed through an activation function, typically rectified linear unit (ReLU). The final stage of the feature extraction layer group is pooling layer. Pooling operation is the process of extracting a statistically representative value within the kernel size. Its purpose is to provide a 'buffer-like' behavior to slight shift or transformation in the input image. The common method of pooling, namely max pooling, takes the maximum value within the kernel window. This set of feature extraction layers (i.e., convolution $\rightarrow$ activation function $\rightarrow$ pooling) may be repeated as many times as required before proceeding to the classification stage.

After the feature extraction, classification is done by passing the output of the convolution layers to a series of fully connected layers. At the interface between the convolution and classification, flattening is usually required since images are handled as 2D tensors in the convolution stage but fully connected layers handle 1D vectors. The final layer has a dimension equal to the number of classes in the classification labels.

III. Methodology

A building damage classification model using CNN is set up and trained for this study. The training, validation, and testing is done on a publicly available post-disaster aerial imagery dataset. Description of the dataset, model, and training process is given below.

A. The xBD dataset: A post-disaster damage dataset

The xBD dataset [2], [10] is a dataset of aerial images taken after a range of natural disaster around the world labeled according to a four-scale damage level. Although the dataset contains several disaster types, hurricanes Matthew (2016), Hervey (2017), Michael (2018), Florence (2018), and are chosen for this study. A sample pair of before and after disaster aerial images are shown in Fig. 1. Each individual image contains multiple buildings marked with polygons. The dataset provides the damage level observed in each building based on the definitions given in Table I.

Fig. 1. Sample aerial images showing before (left) and after (right) disaster.

TABLE I
DEFINITION OF BUILDING DAMAGE LEVELS USED TO LABEL POST-DISASTER IMAGES [2].

Damage class	Description
0 (no damage)	Undisturbed. No sign of wind damage observed.
1 (minor damage)	Building partially damaged. Roof elements missing.
2 (major damage)	Partial wall or roof collapse.
3 (destroyed)	Completely collapsed, or missing from the location.

B. Data processing

Once the selected labeled images are extracted for the four hurricanes, they are processed into a form suitable to feed to the model. The process is demonstrated in Fig. 2. All the provided images are $1024 \times 1024 \times 3$ pixels in size. First, the background in the large image is masked using the provided building polygons as boundaries. The masking is done by turning the pixels outside the polygons off. After the masking, the masked image is broken down into individual buildings with the extents rectangle touching the building polygon. This process results in a variety of rectangular shapes and sizes. To simplify the data management inside the model, all the individual buildings are resized to $128 \times 128 \times 3$ pixels. Sample buildings after this processing are shown at the bottom of Fig. 2 with their corresponding damage levels. From the four hurricanes, a total of about twenty thousand individual buildings with damage level labels are extracted. After the extraction, they are batched and split into training (80%), validation (10%), and testing (10%). The distribution of each damage class in all three sets is given in Table II.

The processed data is arranged in batches to improve the memory usage and training rate. Four different types of random transformations are applied whenever a batch of images is retrieved. Random resizing of scale ranging from 0.8 to 1.0 times the image size and cropping is the first transformation. Another transformation is random rotation of up to 15^0. The third is random color jitter which is randomly changing the brightness, saturation, and other color related properties of the image. The final transformation is random horizontal flip. The

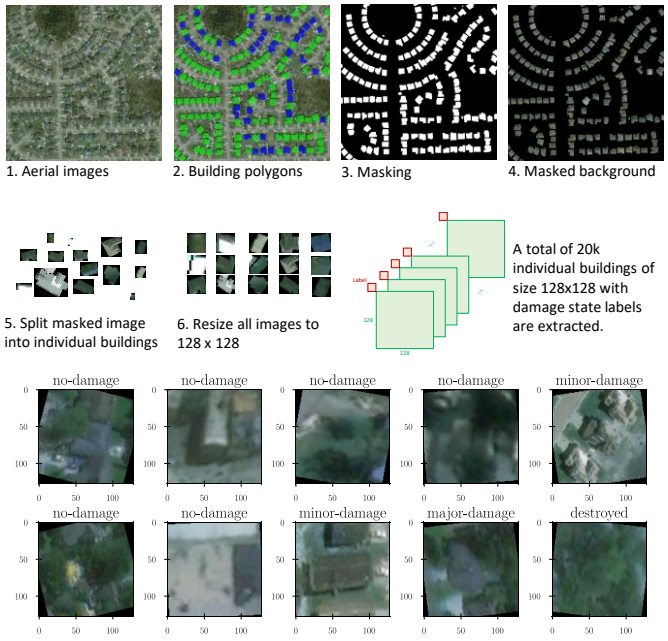

Fig. 2. Data extraction for training, validation, and testing from aerial images.

purpose of all these transformations is to reduce the chance of overfitting, especially with smaller dataset. Their application enables to generate new features without having to add training data. For our model, we added these after preliminary training showed overfitting early in the process.

TABLE II
DISTRIBUTION OF DAMAGE CLASSES IN TRAINING, VALIDATION, AND TEST SETS.

Damage class	Number of buildings		
	Training	Validation	Testing
0 (no damage)	8500	1062	1076
1 (minor damage)	3715	461	447
2 (major damage)	2801	347	352
3 (destroyed)	847	113	108

C. Building damage classification model

The building damage classification model is built using CNN architecture as shown in Fig. 3. It has three sets of convolution based feature extraction layers and four fully connected layers after that for the classification purpose. Each of the convolution based feature extraction layer set includes 2D convolution layer followed by ReLU activation function, then 2D batch normalization, and finish with 2D max pooling. The convolution layer has a kernel size of 5×5 in the first set of feature extraction layers and 3×3 in the other two. The stride The kernel size for the max pooling layers is 2×2 in all three sets. After running preliminary training, early overfitting and instability was observed. In the final model, the batch normalization layers were added to improve the training speed and stability.

The second part of the model after the feature extraction convolution and pooling layers is classification layer comprised of four fully connected linear layers. Since the feature extraction layers are all 2D, flattening of the tensors is done before passing their output to the classification part. ReLU activation function is applied after each fully connected linear layer including the last one to the output. The output is returned as a one-hot encoded vector whereby the maximum value represents the predicted damage class.

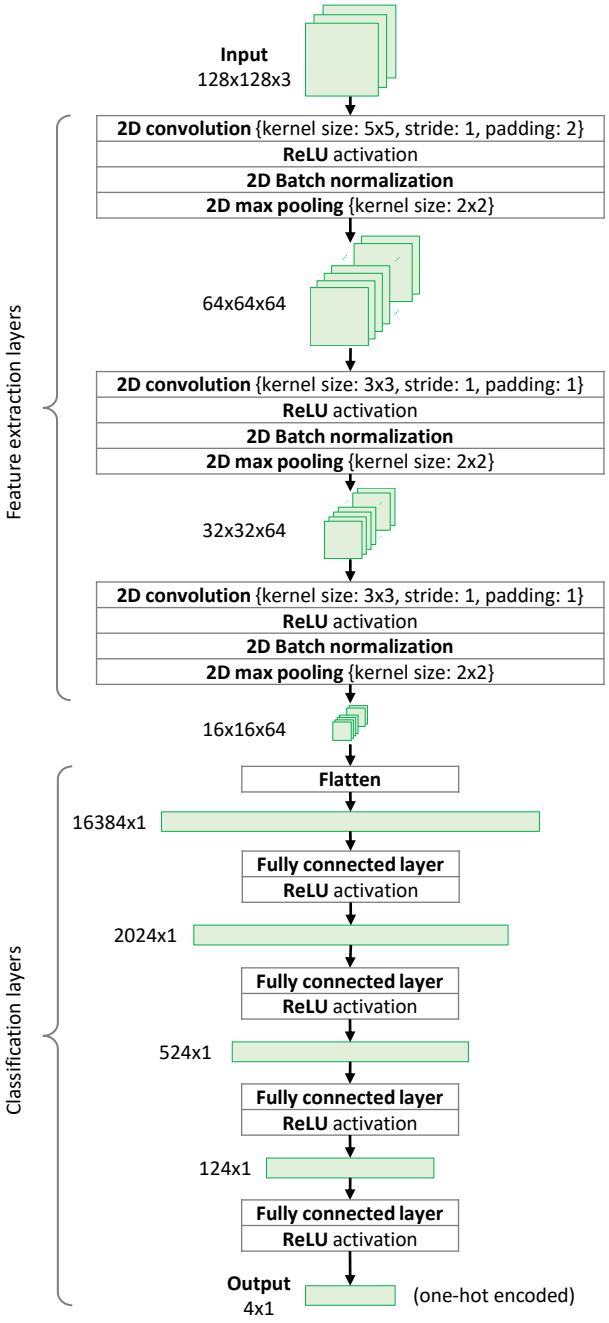

Fig. 3. Schematic diagram of CNN model for building damage classification.

D. Loss function and evaluation criteria

The loss function defining the error in predicted value y relative to true value x used for training and validation is a weighted cross entropy loss function defined as [11]:

$$L = l(x, y) = \frac{1}{N} \sum_{n=1}^{N} l_n \tag{1}$$

where:

$$l_n = -\sum_{c=1}^{C} w_c \log \frac{\exp(x_{n,c})}{\exp\left(\sum_{i=1}^{C} x_{n,i}\right)} y_{n,c} \tag{2}$$

and C is the number of classes, N is the batch size, and w_c is the weight of each class in the dataset. This loss function is commonly applied to multi-class classification as in the present case.

For the evaluation with the testing set, weighted F1 score defined in Eq. 3 is used in addition to the cross entropy loss function [12].

$$\text{F1} = \frac{1}{C} \sum_{c=1}^{C} w_c \cdot \text{F1}_c \tag{3}$$

where:

$$\text{F1}_c = 2 \cdot \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{4}$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6}$$

and C is the number of classes with w_c proportion in the dataset. This metric is chosen because the dataset has biased proportion of classes that is a reflection of the realistic scenario that in most disasters the highest damage classes are rare. The weight factors w_c for the four damage classes are computed according to their distribution in each dataset given in Table II. Definition of confusion matrix is given in Table III. Previous studies [2], [5] have also used F1 score to evaluate their deep learning models for building damage classification.

TABLE III
DEFINITION OF CONFUSION MATRIX FOR A SINGLE CLASS c RELATIVE TO THE REST.

		Prediction y	
		Class c	**Not class** c
Truth x	**Class** c	true positive (TP)	false negative (FN)
	Not class c	false positive (FP)	true negative (TN)

The third evaluation criterion is the percentage balanced accuracy score defined as:

$$\text{accuracy} = \frac{1}{C} \sum_{c=1}^{C} w_c \cdot \text{accuracy}_c \cdot 100\% \tag{7}$$

where:

$$\text{accuracy}_c = \frac{1}{2} \left(\frac{\text{TP}}{\text{TP} + \text{FN}} + \frac{\text{TN}}{\text{TN} + \text{FP}} \right) \tag{8}$$

E. Training, validation, and tuning

The training and validation algorithm is given in Fig. 5. The training process starts by taking in the initialized model, loss function, optimizer, maximum number of epochs, and training and validation datasets. Since the input data is fed in batches, the loop is nested with epochs. First, the optimizer is initialized per each run through a batch. Then, prediction on the next batch of training set is inferred with the CNN model and the loss is calculated. The loss is expressed using the weighted F1 score. The calculated loss is then back propagated to update the model parameters accordingly. Then the optimization is executed. If there is more batch in the training set, this process is repeated until all batches have passed through the model. Note that every time a batch is retrieved, the random transformations explained in §III-B.

Once the model is trained with all the batches in the training set, it is used to predict the validation sets as well. The same loss function is applied here too. After the validation, the whole training and validation process is repeated until the maximum number of epochs is reached. At the end, the training and validation loss values per epoch are returned along with the trained model.

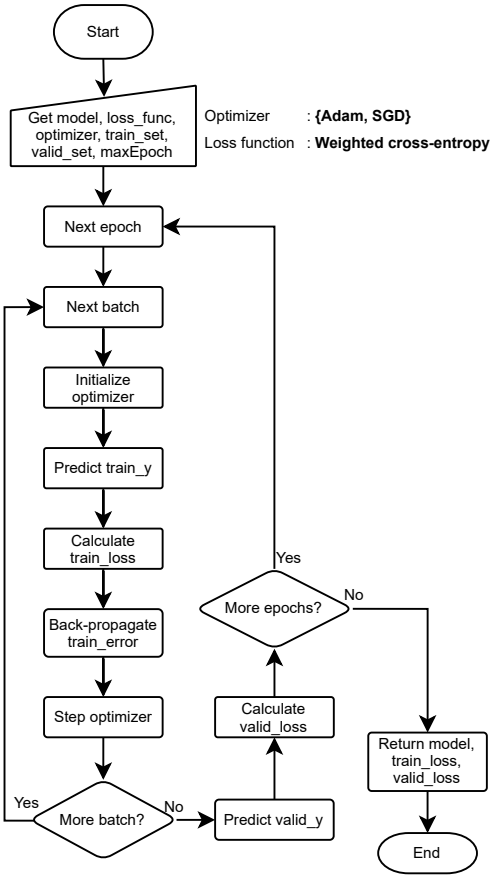

Fig. 4. Flowchart for training and validation algorithm of the CNN model with data input in batches.

This training and validation is a process applied for a single model hyperparameter set. To get the best configuration, the

model is tuned on three hyperparameters: optimizer, learning rate, and epoch. Table IV shows the list of models with varying hyperparameter values. All the training and tuning is done on Graham supercomputer of Compute Canada with node specification is 32 cores of 2 × Intel E5-2683 v4 Broadwell @ 2.1GHz CPU and 2 × NVIDIA P100 Pascal (12 GB HBM2 memory) GPU cores. On average, it took about 0.4 wall-clock hours to train one epoch.

TABLE IV
HYPERPARAMETERS USED IN MODEL TUNING AND THEIR RANGES.

Model ID	Optimizer	Learning rate	Epoch
01	Adam	1e-4	100
02	SGD	1e-4	100
03	Adam	1e-5	100
04	Adam	5e-5	100
05	Adam	1e-4	100
06	Adam	1e-4	10
07	Adam	1e-4	25
08	Adam	1e-4	50

IV. RESULTS AND DISCUSSION

The results presented in this section are combination of preliminary and final models to discuss the training and tuning process as well as the final model.

A. Training and validation results

A sample preliminary training and validation result is given in Fig. 5c and d. The preliminary model did not have the batch normalization layers shown in Fig. 3. The loss and accuracy with increasing epoch is shown. The training loss generally reduces with any significant reduces (or accuracy increases) in epoch. However, the validation loss starts to flatten and eventually start to climb after some optimal epoch number. In the current sample, the optimal epoch count is approximately in the range of 10 to 20. Further continuation in training indicates signs of overfitting. The optimal epoch size changes with change in other parameters like learning rate and type of optimizer. This is why the epoch is included as a hyperparameter.

After observing the slow training in the preliminary model (Fig. 5a and b), the batch normalization layers are added to the model (Fig. 5c and d). This enables the training to be stopped early saving computational time and avoiding a potential source of overfitting at the same time.

B. Tuning results

Once the initial high-level experimentation with model features like batch normalization and image transformation is done, the tuning is conducted. The results of the tuning process are shown in Fig. 6. The evaluation criteria used for model selection is F1-score. A total of eight models are evaluated to pick the best among them. Since for F1-score, the higher, the better, model 03 is the best model with overall F1-score of 0.741.

Considering the limited range of hyperparameters included in this study, the model would potentially improve more with further inclusion of hyperparameters. Other potentially important hyperparameters, such as kernel sizes, number of convolution layers, and number of classification layers, could improve the model further. This task is postponed for future part of the study to focus on other aspects of the model in the limited time and resources.

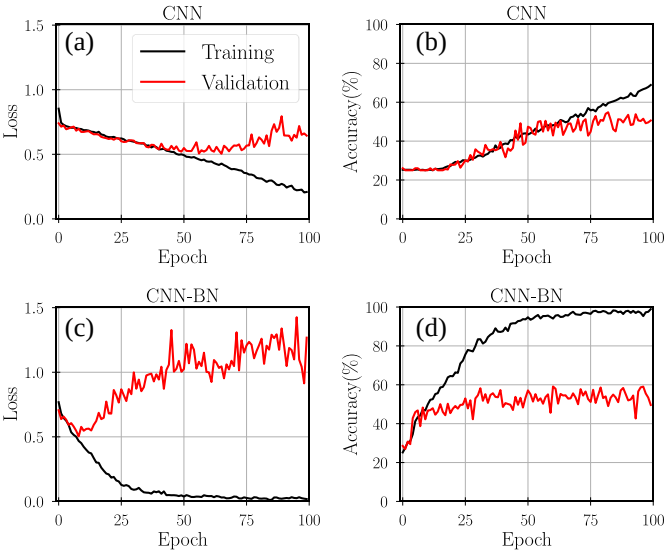

Fig. 5. Training and validation (a) loss and (b) accuracy from CNN without batch normalization in comparison to (c) loss and (d) accuracy from model with batch normalization CNN-BN

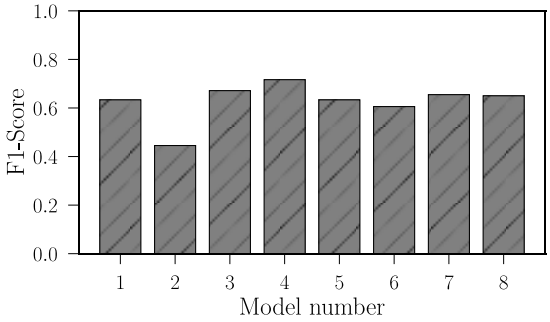

Fig. 6. Comparison of training and validation for (a)

C. Tuned model results and comparison to literature

Based on the tuning results, Model 03 has been selected as the best model. The detailed evaluation results of this model are given in Table V. The F1-score, precision, and recall per each damage class and overall are given in the table. Recall that for all three measures, the best value is one. The 'no damage' class shows the highest in all three measures. The bias present in the dataset has been mitigated by the use of loss and evaluation criteria that account for biased data. However, there could be a secondary bias in features among classes. For a given image, features describing 'no damage' state are more likely to be present than indicator features of the other damaged states. The 'destroyed' class show the least values

of all three measures. The other two classes are close to each other which is likely because of the close physical appearance of the two classes as discussed by ref. [2] as well.

The current best model (Model 03) are compared to studies that used the same dataset for the same building damage classification problem as shown in Table VI. The model by ref. [2] is the base model provided with the dataset that uses transfer learning based on ResNet50. With F1-score of 0.265, its performance is not very good compared to other studies that used the dataset. Another study by ref. [3] also applied transfer learning using U-Net with attention mechanism. Their model has F1-score of 0.792. The third study we compared is [5], that also used transfer learning with GoogLeNet and ResNet. They attained F1-score of 0.868. The current model has F1-score of 0.741 which can be considered to be reasonably good compared to the references.

TABLE V
EVALUATION RESULTS FOR THE CURRENT TUNED MODEL (MODEL 03).

Damage class	**F1-score**	**Precision**	**Recall**
0 (no damage)	0.809	0.881	0.842
1 (minor damage)	0.641	0.559	0.589
2 (major damage)	0.702	0.672	0.676
3 (destroyed)	0.605	0.406	0.463
Overall	**0.741**	**0.746**	**0.735**

TABLE VI
COMPARISON OF CURRENT RESULTS WITH RESULTS FROM LITERATURE THAT USE THE SAME DATASET.

Model	**F1-score**	**Precision**	**Recall**
By ref. [2]	0.265	-	-
By ref. [3]	0.792	-	-
By ref. [5]	0.868	0.888	0.858
Current best	0.741	0.746	0.735

Note: Ref. [2] is the base model provided with the dataset.

V. SUMMARY AND CONCLUSION

A CNN based deep learning classification model is used to identify damage levels in buildings that faced hurricane winds. A publicly available post-disaster aerial imagery dataset is used for training, validation, and testing. The damage scales are defined in four scales from 'no damage' to 'destroyed'. Weighted cross entropy is used as the primary loss function for the training and evaluation is presented with additional measures of F1-score and balanced accuracy. Batch normalization and random image transformations are used to improve the performance of the model. Hyperparameter tuning is done on the optimizer, learning rate, and number of epochs.

The tuned model has F1-score of 0.741, precision of 0.746, and balanced accuracy of 63.9%. Comparing these values to studies that used the same dataset showed that our model performs reasonably well.

Going forward, the current model may be improved by considering transfer learning from existing classification networks, particularly related to remote sensing and similar applications. Consideration of more wind hazard types, such as tornados, and other archetypes would make it more universal.

ACKNOWLEDGMENT

We would like to thank Compute Canada and SHARCNET for providing the HPC platforms with GPU to train and evaluate our models.

SOURCE CODE

Please find a separately attached file for the source code.

REFERENCES

[1] A. B. Smith and R. W. Katz, "US billion-dollar weather and climate disasters: data sources, trends, accuracy and biases," *Nat. Hazards*, vol. 67, no. 2, pp. 387–410, 2013.

[2] R. Gupta, B. Goodman, N. Patel, R. Hosfelt, S. Sajeev, E. Heim, J. Doshi, K. Lucas, H. Choset, and M. Gaston, "Creating XBD: A dataset for assessing building damage from satellite imagery," in *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work.*, pp. 10–17, 2019.

[3] C. Wu, F. Zhang, J. Xia, Y. Xu, G. Li, J. Xie, Z. Du, and R. Liu, "Building damage detection using U-Net with attention mechanism from pre- and post-disaster remote sensing datasets," *Remote Sens.*, vol. 13, no. 5, pp. 1–22, 2021.

[4] J. Thomas, A. Kareem, and K. W. Bowyer, "Automated poststorm damage classification of low-rise building roofing systems using high-resolution aerial imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 7, pp. 3851–3861, 2014.

[5] B. J. Wheeler and H. A. Karimi, "Deep learning-enabled semantic inference of individual building damage magnitude from satellite images," *Algorithms*, vol. 13, no. 8, 2020.

[6] C.-s. Cheng, A. H. Behzadan, and A. Noshadravan, "Deep learning for post-hurricane aerial damage assessment of buildings," *Comput. Civ. Infrastruct. Eng.*, vol. 36, pp. 695–710, 2021.

[7] Y. Pi, N. D. Nath, and A. H. Behzadan, "Convolutional neural networks for object detection in aerial imagery for disaster response and recovery," *Adv. Eng. Informatics*, vol. 43, p. 101009, 2020.

[8] A. Lenjani, C. M. Yeum, S. Dyke, and I. Bilionis, "Automated building image extraction from 360° panoramas for postdisaster evaluation," *Comput. Civ. Infrastruct. Eng.*, vol. 35, no. 3, pp. 241–257, 2020.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[10] Defense Innovation Unit, *xView2: Assess Building Damage*. Web page: https://xview2.org/.

[11] A. Paszke, S. Gross, S. Chintala, and G. Chanan, "PyTorch, Website: https://pytorch.org/," 2021.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.