Deep Learning Based Damage Detection on Post-Hurricane Satellite Imagery

Quoc Dung Cao and Youngjun Choe

***Abstract*—After a hurricane, damage assessment is critical to emergency managers and first responders. To improve the efficiency and accuracy of damage assessment, instead of using windshield survey, we propose to automatically detect damaged buildings using image classification algorithms. The method is applied to the case study of 2017 Hurricane Harvey.**

I. Introduction

When a hurricane makes landfall, situational awareness is one of the most critical needs that emergency managers face before they can respond to the event. To assess the situation and damage, the current practice largely relies on emergency response crews and volunteers to drive around the affected area (also known as windshield survey). Recently, drone-based aerial images and satellite images have started to help improve situational awareness, but the process still relies on human visual inspection. These current approaches are generally time-consuming and unreliable during an evolving disaster.

The proposed algorithm can automatically detect *'Flooded/Damaged Building' vs 'Undamaged Building'* on satellite imagery of an area affected by a hurricane. This could give the stakeholders useful information about the severity of the damage to plan for and organize the necessary resources. This is expected to significantly reduce the time for building situational awareness and responding to hurricane-induced emergencies.

The satellite imagery data used in the paper was captured and preprocessed for orthorectification, atmospheric compensation, and pansharpening from the Greater Houston area before and after Hurricane Harvey in 2017. The damaged buildings were labeled by the volunteers through crowd-sourcing. We then process, filter, and clean the dataset to ensure that it has higher quality and can be learned appropriately by the deep learning algorithm.

Through this paper, other researchers can use the dataset and methodology to study and experiment with different uses of satellite imagery in disaster response. We also hope to provide a pre-trained architecture that achieves satisfactory result. It can facilitate transfer learning either in feature extraction, fine-tuning, or as a baseline to speed up the learning process in future development/events with similar properties.

Quoc Dung Cao and Youngjun Choe are with the Department of Industrial and Systems Engineering, University of Washington, Seattle, WA 98195.

II. Background

Object detection is a ubiquitous topic in computer vision, thanks to the development of convolutional neural network (CNN) [1]. Nevertheless, few studies have investigated machine learning based damage detection on post-hurricane satellite imagery. A small project studied detecting *flooded roads* by comparing pre-event and post-event satellite imagery [2] but the method is not applicable to other types of damages. Two commercial vendors of satellite imagery also separately developed unsupervised algorithms to detect flooded area using spectral signature of impure water (which is not available from the pansharpened satellite images in our data) [3], [4]. Before deep learning era, a method using a pattern recognition template set was applied to detect hurricane damages in *multispectral* images [5]. The method is not applicable to our pansharpened images. More broadly, object detection on satellite imagery is a well-established research area in remote sensing, although the existing studies focus on detecting roads, buildings, trees, vehicles, ships, airplanes, or airports [6], [7].

There are multiple challenges in damage detection. First, the satellite imagery resolution is not as high as the various state-of-the-art datasets commonly used to train neural networks (NNs). Dodge & Karam studied the performance of NNs under quality distortions and highlighted that NNs could be prone to errors in blurry and noisy images [8]. Although our dataset is of relatively high quality, e.g., one of the satellites capturing the imagery is the GeoEye-1, which has 46cm panchromatic resolution [9], it is still far from the resolution in animal or vehicle detection datasets. In fact, the detection task is hard even for human visual inspection. Second, the volunteers' annotation could be erroneous. To limit this, the imagery provider has a proprietary system that computes the agreement score of each sample. In this paper, we ignore this information to gather as many samples as possible and take the labels as ground truth. Third, there are some inconsistencies in image quality. Since the same region can be captured multiple times in different days, the same coordinate may have different quality and orthorectification.

III. Methodology

A. Data description

1) The raw imagery data covering the Greater Houston area was captured in about four thousand strips ($\sim$ 400 million pixels ($\sim$ 1GB) per strip) in different days. Hence, some strips can overlap, leading to some images blacked out at the boundaries. In some days, the images are also covered fully or partially by clouds. Figure 1 shows a typical strip in the data set and Figure 2 shows some examples of low quality images in 128x128 pixels that we choose to discard. Due to the big volume, it is not feasible to store the data in a local computer. The raw imagery data was downloaded and stored on a high performance computing cluster.

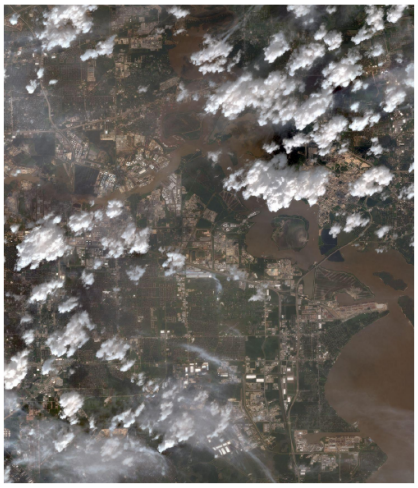


Fig. 1: A typical strip of image

2) The raw data is in geoTIFF format, which allows georeference information to be embedded within a TIFF file.
3) Damaged buildings are annotated with labels and coordinates given in GeoJSON format. Using the coordinates, we extract the images of damaged buildings in JPEG format from the geoTIFF post-event imagery.
4) Undamaged buildings are extracted in JPEG format directly from the geoTIFF pre-event imagery.

B. Damage Annotation

We present here a framework (Figure 3) from raw data to damage annotation. Since there is no readily available data for model training, the first obvious step is to generate the data. We adopt a cropping window approach. Essentially, the building coordinates, which are either easily obtained publicly or available from crowd-sourcing projects, are used as the center of a window. The window is then cropped from the raw satellite imagery to create a data sample. It is not clear what the optimal size of the window should be. Too small windows may limit the background information contained in each sample, whereas too large ones may introduce unnecessary noise. We keep the window size as a tuning hyper-parameter in the model. A few sizes are considered such as 400x400, 128x128, 64x64, and 32x32. The cropped images are then manually filtered to ensure the high quality of the dataset. To let the model generalized well, we only discard obviously flawed images, as shown in Figure 2. The clean images are then split into training, validation, and test sets and fed to a convolutional neural network for damage annotation as illustrated in Figure 3. Validation accuracy is monitored to tune the necessary hyper-parameters and the window size.

(a) Blacked out partially (b) Covered by cloud partially

(c) Covered by cloud mostly (d) Covered by cloud totally

Fig. 2: Examples of 128x128-pixel low quality images

C. Data Processing

As mentioned above, the data generation starts from a building coordinate. Since there are many raw geoTIFF files containing the same coordinates, there are many duplicate images with different quality. This can potentially inflate the prediction accuracy as the same coordinate may both appear in the training and test sets. We maintain an unordered set of the available coordinates and make sure each coordinate yields a unique, "good-quality" image in the dataset. Here, the definition of "good-quality" is subjective so the process is semi-automated. We first automatically discard the totally blacked out images for each coordinates, and keep the first images we encounter that is not totally black.

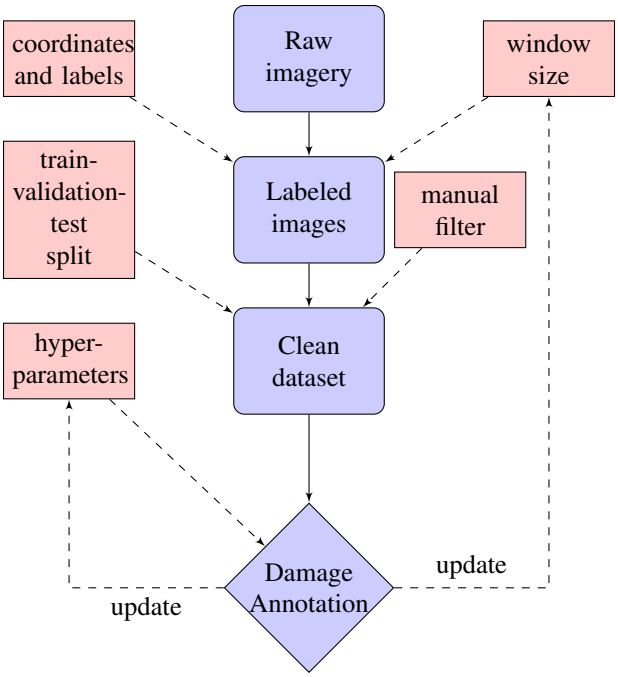
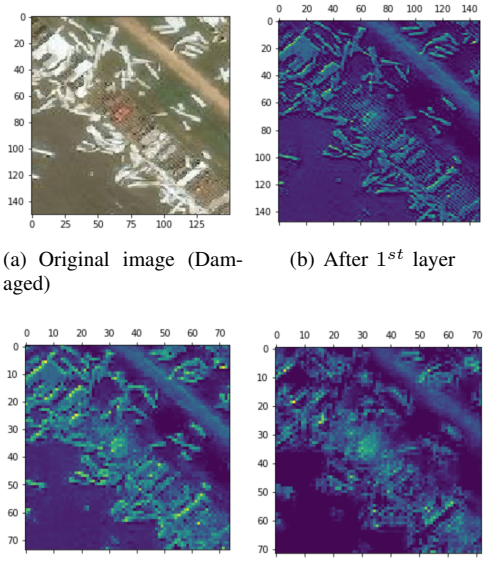


Fig. 3: Damage annotation framework



(a) Original image (Damaged) (b) After 1^{st} layer

(c) After 2^{nd} layer (d) After 3^{rd} layer

Fig. 4: Information flow within one filter

The remaining images are manually filtered to eliminate images that are partially black, and/or covered by clouds.

D. Data Featurization

Since we control the window size through physical distance, there could be round off errors when converting distance to the number of pixels. When we featurize the images, we project them into the same feature dimension. For instance, 128x128 images are projected into 150x150 dimension. The images are then fed through a CNN to further extract the right set of features, such as edge extraction in Figure **??**.

How to construct the most suitable CNN architecture is an ongoing research problem. The practice is usually starting with an existing architecture and fine tune further from there. We experiment with a well-known architecture, VGG-16 [10], and modify the first layer to suit our input dimension. VGG-16 can perform extremely well in the ImageNet dataset for object detection.

However, realizing the crucial differences between common objects detection and damage building annotation tasks, we also build our own network from scratch with proper hyper-parameters. Our basis for determining the size and depth of a customized network is to monitor the information flow through the network and stop appropriately when there are too many "dead" filters. Due to the nature of the rectified linear unit (ReLU) which is defined as $max(0, x)$, there will be many hard 0 in the hidden layer. Although sparsity in the layer will promote the model to generalize better, it can potentially cause the problem to gradient computation at 0 and hurt performance [11], [12]. We see that in Figure 5 after 4 hidden layers, about 30% of the filters are "dead" and will not carry further information to the subsequent layers. This is a significant stopping criterion since we can avoid a deep network such as VGG-16 to save the computational time and safeguard satisfactory information flow in the network at the same time.

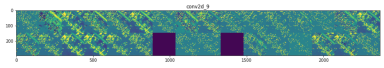
E. Image Classification

Due to the presence of flawed images in the *Damaged* and *Undamaged* categories, we experience an unbalanced dataset with the majority class being *Damaged*. As a result, the following training, validation, test splitting method is adopted. We keep the training and validation sets balanced and leave the remaining data to construct 2 test sets, a balanced and an unbalanced (with a ratio of 1:8) sets.

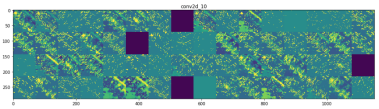
The first performance metric is the classification accuracy. Based on the unbalanced test set, the baseline performance is determined by annotating all buildings as the majority class *Damaged* with $\frac{8}{9} = 88.89\%$ accuracy. To be comprehensive, we also monitor the area under the receiver operating characteristic curve (AUC).

IV. Implementation and Result

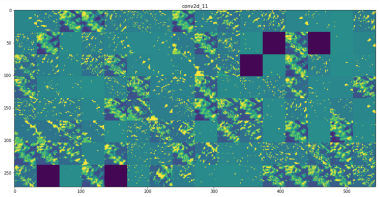
We train the neural network through the *Keras* library with TensorFlow backend with a single NVIDIA K80 Tesla GPU with 64GB memory on a quad-core CPU machine. The network weights are initialized through



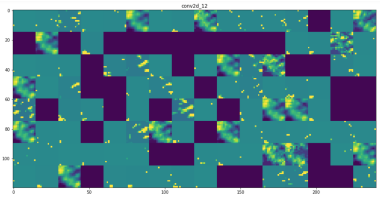
(a) After 1^{st} layer



(b) After 2^{nd} layer



(c) After 3^{rd} layer



(d) After 4^{th} layer

Fig. 5: Information flow in all filters after each layer

Xavier initializer [13]. The mini batch size for stochastic gradient descent optimizer is either 20 or 32.

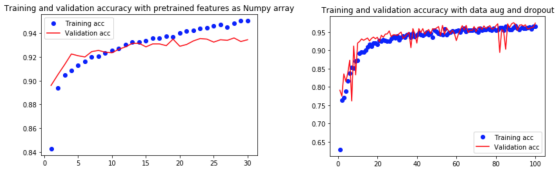
After cleaning and manual filtering, we are left with 14,284 positive samples (*Damaged*) and 7,209 negative samples (*Undamaged*) of unique coordinates. 5,000 samples of each class are in the training set. 1,000 samples of each class are in the validation set. The rest of the data are reserved to form the test sets.

Since it is computationally costly to train the CNN repeatedly, we do not tune the hyper-parameters through a full grid search or full cross-validation. Only some reasonable combinations of the hyper-parameters are considered. Among the parameters, window size is truly a challenge. We do not try all the sizes with all hyper-parameters. We implement a simple model with all the window sizes and find that 128x128 window yields an ideal result.

We also implement a logistic regression (LR) on the featurized data to see how it compares to a densely connected layer. LR under-performs in most cases but also achieves quite good accuracy. This illustrates that the image featurization through the network is very crucial to extract good features such that a simple algorithm can perform well enough on this data.

For activation functions in CNN, a rectified linear unit (ReLU) is a common choice, thanks to its simplicity in gradient computation and prevention of vanishing gradient. As seen in Figure 5, clamping the activation at 0 could potentially cause a lot of filters to be dead. Therefore, we also consider using a leaky ReLU activation, which is defined as $max(\alpha x, x)$, with $\alpha \ll 1$. We pick $\alpha = 0.1$ in this case, based on the survey in [12]. However, leaky ReLU does not improve the accuracy very much.

To counter over-fitting, which is a recurrent problem of deep learning, we also adopt data augmentation in the training set through random rotation, horizontal flip, vertical and horizontal shift, shear, and zoom. This can effectively increase the number of training samples to ensure more generalization to achieve better validation and test accuracy (We do not perform data augmentation in the validation and test set). Further more, we also employ 50% drop out and L2 regularization with $\lambda = 10^{-6}$ in the densely connected layer. These measures are shown to fight over-fitting effectively and significantly improve the validation accuracy in Figure 6.



(a) Over-fitting happens after a few epochs (b) Little sign of over-fitting

Fig. 6: Prevent over-fitting using data augmentation, drop-out, and regularization

As mentioned in Section III-D, we consider using a pre-built architecture VGG-16 (transfer learning) and building a fresh network. In Figure 7, we see that using a deeper and larger network we can achieve a high level accuracy earlier but over-fitting happens after a first few epochs. Our simpler network can facilitate learning gradually, achieve better accuracy, and take less time to train.

(a) Transfer learning (b) Custom network

Fig. 7: Comparison between using a pre-built network and our custom network

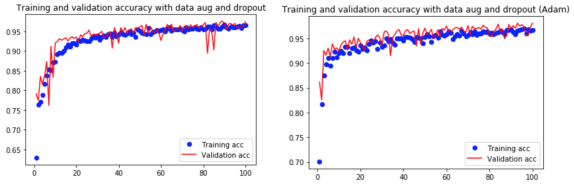
We use two adaptive, momentum based optimizers RMSprop and Adam [14] with initial learning rate of 10^{-4}. Adam generally leads to about 1% higher valida-

Model	Val. Acc.	Test Acc. (Balanced)	Test Acc. (Unbal.)
CNN	95.8%	94.69%	95.47%
Leaky CNN	96.1%	94.79%	95.27%
CNN + DA + DO	97.44%	96.44%	96.56%
CNN + DA + DO (Adam)	**98.06%**	**97.29%**	**97.08%**
Transfer + DO	93.45%	92.8%	92.8%
Transfer + DA + DO	91.1%	88.49%	85.99%
LR + L2 = 1	93.55%	92.2%	91.45%
Transfer + DA + FDO	96.5%	95.34%	95.73%
Leaky+Transfer + DA + FDO +L2	96.13%	95.59%	95.68%
Leaky+ Transfer + DA + FDO +L2 (Adam)	97.5%	96.19%	96.21%

Legend: CNN: Convolutional Neural Network; Leaky: Leaky ReLU, else default is ReLU; DA: Data Augmentation; LR: Logistic Regression; L2: L2 regularization; (Adam): Adam optimizer, else default is RMSprop optimizer; DO: 50% drop out in the densely connected layer; FDO: Full drop out, i.e 25% drop out after every max pooling layer and 50% in the densely connected layer; Transfer: Transfer learning using VGG-16 architecture

TABLE I
MODEL PERFORMANCE

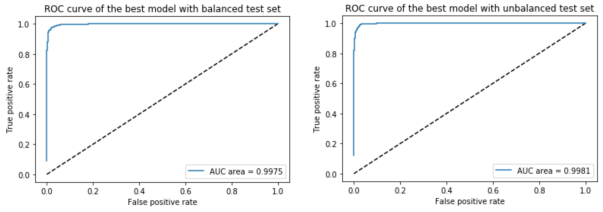
tion accuracy and it can be seen that Adam leads to less noisy learning (Figure 8).



(a) RMSprop (b) Adam

Fig. 8: Comparison between using RMSprop and Adam optimizers

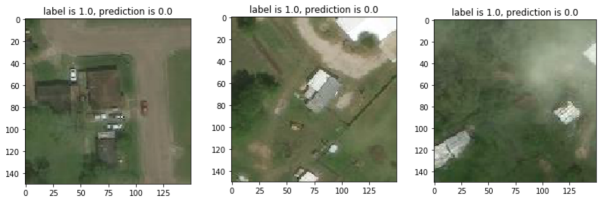
Table I demonstrates the performance of various models. The best performing model is our fresh model with data augmentation and drop out using Adam optimizer, which can achieve 97.08% accuracy on the unbalanced test set. The AUC metric is also computed and shows a satisfying result of 99.8% on the unbalanced test set.



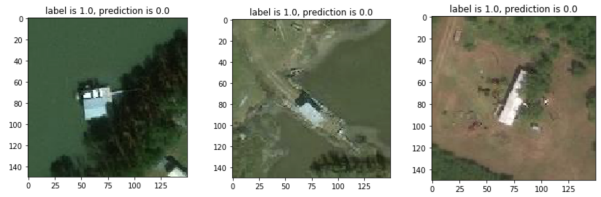
(a) AUC of balanced test set (b) AUC of unbalanced test set

Fig. 9: AUC for balanced and unbalanced test sets

Although the result is satisfactory, we also look at a few typical cases where the algorithm makes wrong classification to see if any intuition can be derived. Figure 10 shows some of the false positive cases. We hypothesize that the algorithm could predict the damage through flood and debris detection. Under such hypothesis, the cars in the center of Figure 10(a), the lake water in Figure 10(b), the cloud covering the house in Figure 10(c), the tree covering the roof in Figure 10(f) can potentially mislead the model. For the false negative cases in Figure 11, it is harder to make sense out of the prediction. Even through visual inspection, we cannot see Figures 11(a)(b) as being damaged. Figures 11(e)(f) are clearly damaged but the algorithm misses them.

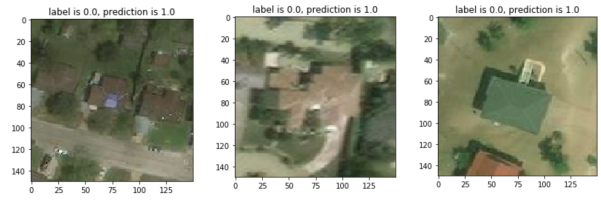


(a) (b) (c)

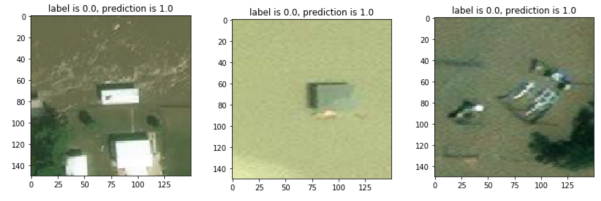


(d) (e) (f)

Fig. 10: False positive examples (label is Undamaged, prediction is Damaged)



(a) (b) (c)



(d) (e) (f)

Fig. 11: False negative examples (label is Damaged, prediction is Undamaged)

V. Conclusion and Future Research

We demonstrated that through deep learning, automatic detection of damaged buildings can be done satisfactorily. Although our data can be specific to the geographical condition and building properties in the Greater Houston area during Hurricane Harvey, the model can be further improved and generalized to other future disaster events in other regions if we can collect more positives samples from other past events and negative samples from other areas.

For faster disaster response, we need a model that can work with low quality images generated on a particular day, especially the hurricane event date, which can be covered by cloud or imperfectly orthorectified. We will further investigate the model to see if it can be robust against such noise and distortion to reduce the amount of manual processing.

Since the positive samples are limited and valuable, we hope to further investigate how to save the samples that are partially blacked out through boundary mirror and enhance contrast to cloud covered samples.

Through the inspection of false positive cases, there could be a link to pixel level classification to segment different damage types, although this requires a massive effort to label different damage shapes and types.

We also wish to expand the current research to road damage annotation which could help plan effective transportation routes of food, medical aid, or energy to the disaster victims.

Acknowledgement

We would like to thank DigitalGlobe for data sharing through their Open Data Program. We also thank Amy Xu, Aryton Tediarjo, Daniel Colina, Dengxian Yang, Mary Barnes, Nick Monsees, Ty Good, Xiaoyan Peng, Xuejiao Li, Yu-Ting Chen, Zach McCauley, Zechariah Cheung, and Zhanlin Liu in the Disaster Data Science Lab at the University of Washington, Seattle for their help with data collection and processing.

References

[1] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, Contour and Grouping in Computer Vision*. London, UK, UK: Springer-Verlag, 1999, pp. 319–. [Online]. Available: http://dl.acm.org/citation.cfm?id=646469.691875

[2] W. Jack, "Road inspector using neural network," https://github.com/jackkwok/neural-road-inspector, 2017.

[3] "Anatomy of a catastrophe," https://www.planet.com/insights/anatomy-of-a-catastrophe/, 2017.

[4] "Unsupervised flood mapping," http://gbdxstories.digitalglobe.com/flood-water/, 2017.

[5] F. H. . Y. J. Barnes, C. F., "Hurricane disaster assessments with image-driven data mining in high-resolution satellite imagery."

[6] G. Cheng and J. Han, "A survey on object detection in optical remote sensing images," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 117, pp. 11 – 28, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0924271616300144

[7] L. Zhang, L. Zhang, and B. Du, "Deep learning for remote sensing data: A technical tutorial on the state of the art," *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, no. 2, pp. 22–40, June 2016.

[8] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*, June 2016, pp. 1–6.

[9] "GeoEye-1 satellite sensor," https://www.satimagingcorp.com/satellite-sensors/geoeye-1/.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Computing Research Repository*, vol. abs/1409.1556, 2014.

[11] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudk, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 315–323. [Online]. Available: http://proceedings.mlr.press/v15/glorot11a.html

[12] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *Computing Research Repository*, vol. abs/1505.00853, 2015. [Online]. Available: http://arxiv.org/abs/1505.00853

[13] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. [Online]. Available: http://proceedings.mlr.press/v9/glorot10a.html

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Computing Research Repository*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980