sensors

MDPI

Article

Crack Detection in Images of Masonry Using CNNs

Mitchell J. Hallee [1], Rebecca K. Napolitano [2], Wesley F. Reinhart [3,4] and Branko Glisic [1,*]

[1] Department of Civil and Environmental Engineering, Princeton University, Princeton, NJ 08544, USA; mhallee@alumni.princeton.edu
[2] Department of Architectural Engineering, Pennsylvania State University, University Park, PA 16802, USA; nap@psu.edu
[3] Department of Materials Science and Engineering, Pennsylvania State University, University Park, PA 16802, USA; reinhart@psu.edu
[4] Institute for Computational and Data Sciences, Pennsylvania State University, University Park, PA 16802, USA
* Correspondence: bglisic@princeton.edu

Abstract: While there is a significant body of research on crack detection by computer vision methods in concrete and asphalt, less attention has been given to masonry. We train a convolutional neural network (CNN) on images of brick walls built in a laboratory environment and test its ability to detect cracks in images of brick-and-mortar structures both in the laboratory and on real-world images taken from the internet. We also compare the performance of the CNN to a variety of simpler classifiers operating on handcrafted features. We find that the CNN performed better on the domain adaptation from laboratory to real-world images than these simple models. However, we also find that performance is significantly better in performing the reverse domain adaptation task, where the simple classifiers are trained on real-world images and tested on the laboratory images. This work demonstrates the ability to detect cracks in images of masonry using a variety of machine learning methods and provides guidance for improving the reliability of such models when performing domain adaptation for crack detection in masonry.

Keywords: computer vision; crack detection; structural health monitoring; masonry; machine learning; convolutional neural network

Citation: Hallee, M.J.; Napolitano, R.K.; Reinhart, W.F.; Glisic, B. Crack Detection in Images of Masonry Using CNNs. *Sensors* **2021**, *21*, 4929. https://doi.org/10.3390/s21144929

Academic Editors: Piotr Kohut, Alessandro Sabato, Adam Martowicz and Krzysztof Holak

Received: 1 June 2021
Accepted: 9 July 2021
Published: 20 July 2021

1. Introduction

Masonry construction is common in both historical and contemporary architecture [1–3]. Furthermore, there has been a surge of interest in using masonry for sustainable infrastructure in the future [4–8]. One of the reasons masonry construction achieves a long service lifetime is its ability to be incrementally repaired; the sacrificial mortar and the modular nature of individual masonry blocks means it is less expensive to maintain than monolithic materials like concrete slabs. Consequently, there are a myriad of ways these structures can incur damage over their lifetime. Masonry structures are susceptible to cracking due to thermal stress from freezing and thawing cycles [9–11] or incompatible material adjacency [12], hydroscopic stress from precipitation or rising damp [13,14], as well as mechanical stress from settlement [15–18] or earthquakes [19]. Unreinforced masonry is typically the most vulnerable type of building material to earthquake damage according to the U.S. Federal Emergency Management Agency [20].

Traditionally, masonry facade inspections have been performed through a combination of techniques including: ground-level inspections [21]; tactile inspections using cherry pickers, scaffolding, or ropes [22]; and drone-based inspections [23,24]. These techniques are costly and time consuming because of the mobilization requirements. Even approaches using drone-captured images can be time-consuming since presently a human must manually inspect them to ascertain the state of the structure. The goal of this research is to develop software to automate the crack classification process in a way that is

flexible enough to account for variation in brick size and shape as well as variability introduced between structures, cameras, and lighting conditions (i.e., characteristic of image collections found in an internet search). An inspection methodology based on automated image-based classification algorithm would save time and money compared to traditional approaches [25]. It could also be very useful for time-sensitive assessments following natural disasters, when damage is widespread and resources are strained [26,27].

Computer-vision-based crack detection has been explored in the context of other materials including road surfaces [28–34], steel [35–38], concrete [39,40]. These have relied on traditional computer vision techniques including filters [31], transforms [39], and edge detection schemes [23,35,41,42]. More recently, CNNs have gained popularity [33,34,43]. One benefit of using CNNs is that the selection of features which are relevant for the desired categorization is done automatically by the CNN, for increased speed and performance compared to human selection [44].

CNNs were recently used to detect and localize crack damage in concrete structures [45]. This work compared not only existing network architectures but also compared a specially constructed architecture to pretrained architectures including VGG-16, VGG-19, ResNet-50, and Inception V3 models. It was found that many of the pretrained architectures were not able to perform at the level of the specifically constructed network, and their prediction time was much slower than the specially constructed CNN. Similarly, it has been found that careful tuning of hyperparameters in shallow networks can yield similar results to deeper networks in less time [46]. This particularly is useful in applications for UAV where real-time decision making about where to fly the UAV depends on damage detection and localization. There have been many studies examining how region-based CNNs can augment current work in concrete crack detection [47–49]. Autoencoders have also become increasingly common for crack detection tasks [50–52].

Many recent works are also still employing other sensor modalities (besides optical imagery) for automated crack detection. Contact sensors have been used with NNs to ascertain when cracks were forming on a lighthouse [53]. Wavelet transforms [54] and ultrasonic waves [55,56] have also been analyzed by CNNs for damage detection in concrete.

Algorithmic identification of cracks in mortared-masonry encounters additional challenges not present in homogeneous materials such as concrete. While cracks in concrete can be identified using relatively simplistic methods (support vector machines [57], edge detection methods [39]), non-homogeneous materials such as masonry pose new challenges. As mortar joints can be convex or concave, shadows can be cast both from the mortar joints onto the bricks as well as from the bricks onto the mortar joints. These shadows interfere with health monitoring as they can trigger a false positive when being mistaken for a crack. While false positives are safer than false negatives (which would leave the damage undetected), high rates of false positives can reduce the trustworthiness of the algorithm via alarm fatigue.

Many preliminary implementations of CNNs for masonry crack detection rely on images from a single site [43,58–60] for training and testing, meaning that brick shape, size, and color are all highly regular. Furthermore, many of these first studies[43,61] assume a certain brick size to design a sliding window which excludes joints. Some more sophisticated architectures have been deployed to handle heterogeneity of masonry structures [62], but most studies have focused on a relatively narrow image domain (i.e., images from a single site or a few similar sites).

Here, we explicitly consider images of masonry with heterogeneous composition (i.e., mortar joints). This is an important distinction because cracks in brick walls are frequently in the mortar region, or along the brick-mortar interface. Additionally, while this method initially is tested on images similar to the training data, it is subsequently tested for its applicability to real-world images from the internet showing masonry cracking. The idea of domain adaptation is important to address because industry professionals will not be training a new CNN for each building they are examining, they will be using a tool which is generalized for a diverse array of masonry structures.

We first identify cracks under optimal conditions in a narrow image domain (i.e., consistent materials, uniform lighting, and orthogonal photography [63]). Subsequently, we test the method against images of cracked masonry obtained from Google Image Search and measure its performance. However, a common element missing in prior works is an application of the models to testing data which comes from a different domain than the one it was trained on; any model deployed under realistic circumstances would need to demonstrate a high degree of transferability between structures and conditions. In this work, domain adaptation is specifically considered in the development of the models. Critically, we find that the performance of machine learning models for domain adaptation is greatest when the training data encompasses a broader range of scenarios (i.e., the real-world data), whereas training on a large number of images from a controlled environment results in overfitting.

2. Materials and Methods

2.1. Laboratory-Scale Masonry Walls

Small-scale experimental walls (such as shown in Figure 1) were built using miniature bricks which measured 3.4 × 1.7 × 1.7 cm and were cored with three evenly-spaced holes in the middle. Using small bricks made repeated building and cracking safer and faster. Each wall included 30 to 50 bricks with heights and widths both varying from 6–8 bricks. A total of 100 different bricks were used in the construction of 53 walls.

Figure 1. A sample image of wall built and cracked in the lab.

The bricks were joined together with a non-cementitious mortar so that they could be re-used. Mortar is typically made from a combination of water, sand, and cement. When only removing the cement, the particle size of the sand was too large relative to the miniature bricks (comparable to placing river rocks between full-size bricks). Thus, the particle size needed to be scaled down along with the bricks. An alternative mortar recipe was designed with flour (as the binder), corn meal (as the aggregate) and water. Though not a strong binder, this mortar substitute looked visually similar to real mortar in photographs.

After construction, the walls were allowed to dry for at least two hours until the mortar was set. Walls were cracked manually in many arrangements by pushing, pulling and twisting different sections of the wall until a crack emerged in the mortar. Photos of size 4608 × 3456 pixels were obtained using a mirrorless Nikon D90 digital camera mounted on a tripod with a two second shutter delay. Identifying cracks in later steps required zooming in close onto sections of the photo, so pixel-level sharpness was important. Both of these measures reduced camera motion during the exposure and helped to visibly increase sharpness.

The shooting distance to the walls was variable as to not impact the results. Additionally lighting conditions were varied between the different walls as to not further bias the system. This is visible in Figure 1, where a blue tint is imbued by a blue tarp in the scene.

2.2. Real-World Images

The models were also tested for their ability to classify real-world images after being trained on laboratory images. Pictures of different masonry walls (cracked and uncracked) were obtained using Google Image Search. To ensure that the architecture was tested on a variety of wall typologies, walls with bricks of different size, colors, shapes, level of deterioration (burned, spalling, with effloresence) were utilized. Additionally, variety in the mortar typology (convex, concave) was included to test the efficacy of the developed architecture on diverse wall constructions (see Figures 2 and 3).

Figure 2. Illustration of some images found on Google which were labeled "Cracked".

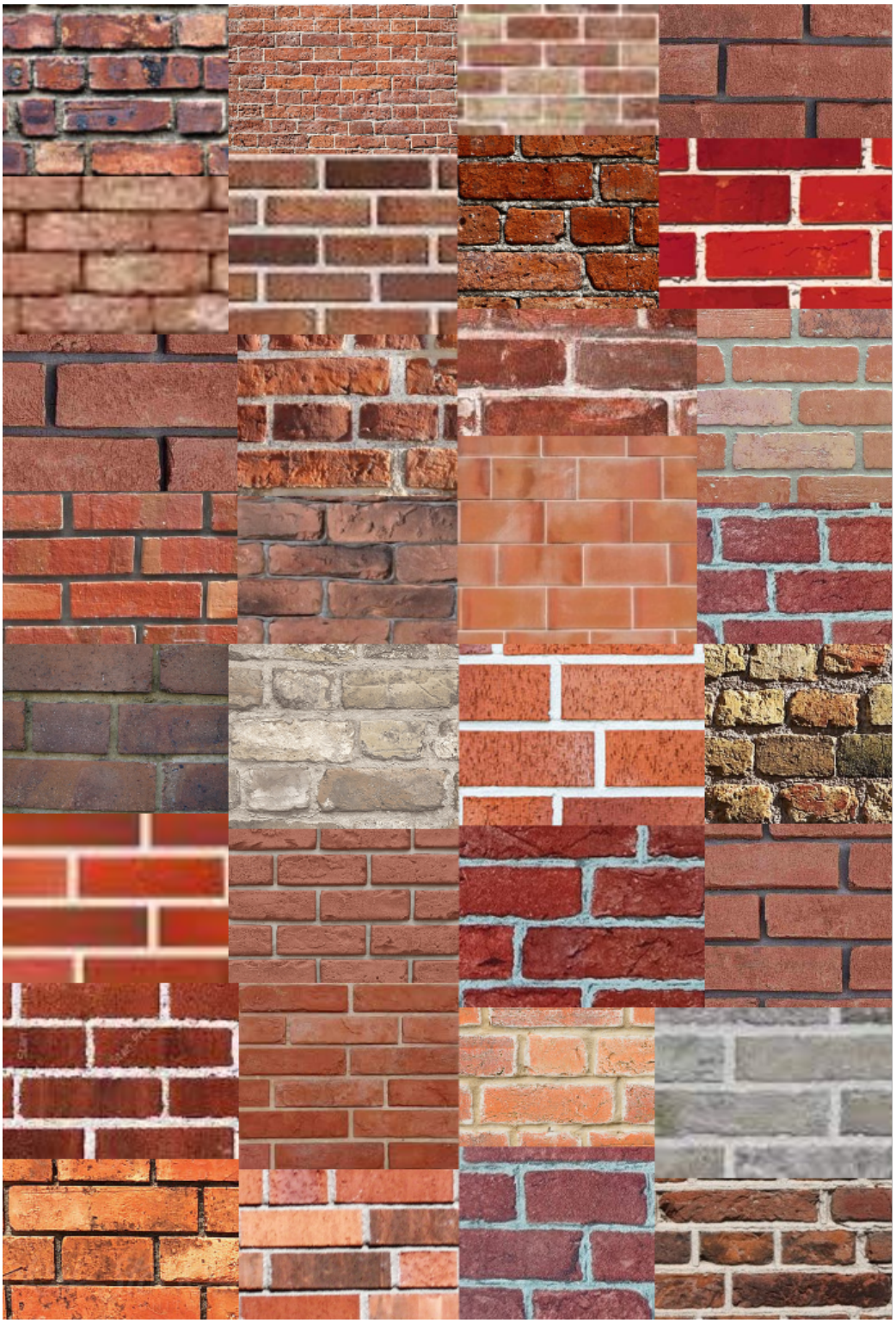

Figure 3. Illustration of some images found on Google which were labeled "Uncracked".

2.3. Image Processing

The images were prepared for the CNN by splitting each 4608 × 3456-pixel image into 48,512 × 512-pixel non-overlapping image patches, as shown in Figure 4. A total of 2542 image patches were labeled manually. The following classes were used:

1. Cracked: Image clearly shows a cracked section of a brick wall.
2. Uncracked: Image clearly shows an uncracked section of a brick wall.

3. Vague: It is unclear whether or not there is a crack in the image. There may be a very small crack, the crack might fall directly on the image edge, or the image might be out of focus.
4. Partial: These images include part of a brick in addition to some of the black background on which the wall was resting when photos were taken.
5. No-bricks: These images include only black background with no bricks.

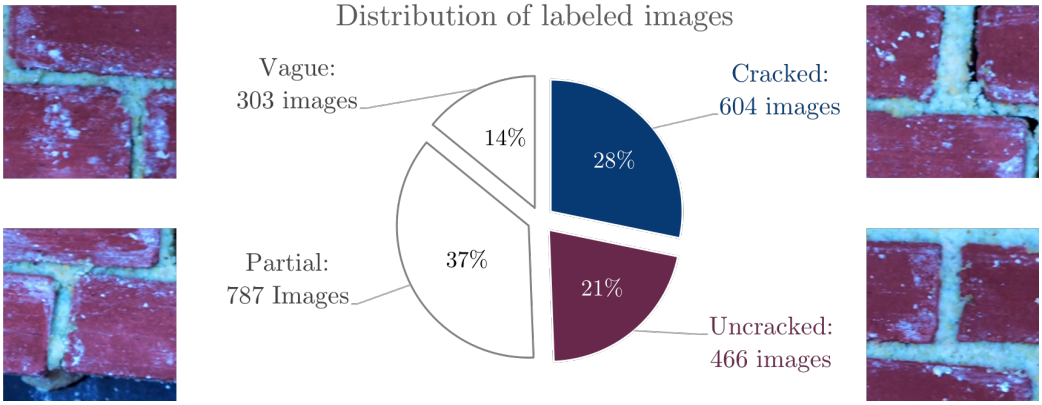

Figure 4. The distribution of images in the data set with representative examples.

Only the Cracked and Uncracked images were used for training. For many of the Vague images, the authors did not agree on whether the image was cracked or not. Similarly, some Partial images had marks which looked like they might be a crack at the edge, which resulted in split votes on whether a crack was present. This yielded a total of 1068 usable images which were further split up into:

- a training set (642 images) on which to train the model,
- a development set (213 images) on which to tune hyperparameters,
- a testing set (213) images on which to test the final results.

Additionally, some cracked images were removed from the training set to yield a 1:1 Cracked:Uncracked ratio, reducing the training set size to 598.

The images taken as 512 × 512 pixel crops were then scaled down to 100 × 100 pixels using bicubic interpolation over 4 × 4 pixel neighborhood. Unlike the initial reduction to 512 × 512, the reduction from 512 × 512 to 100 × 100 did not involve changing the crop of the image, only a slight blurring. This blurring of the images mitigates distinction between images taken under laboratory conditions and real-world images. Based on previous works by the authors where cracks were mapped manually on masonry structures [24,64–66], 150 × 75 pixel patches are sufficient for documentation of individual bricks.

2.4. CNN Model

Based on the results of previous research [44,58–60,62], a CNN is used to classify the images presented in this work. The CNN model was implemented with the Keras deep learning library [67] using the TensorFlow backend. The architecture of Ref. [44] was used as a baseline due to its high success rate in identifying cracks in concrete. Two types of labels are used in the softmax layer to classify the images (Cracked and Uncracked). The overall architecture is summarized in Table 1.

The first modification was designed around a goal of having a 15 × 15 filter to start. This was based on the hypothesis that it would be beneficial to have a filter large enough to see large swaths of cracked material surrounded on both sides by uncracked material. The sizes of other filters were set to keep the height and width of the data large enough to run through all 18 layers. The result has been named Architecture B and is shown in Table 2.

Table 1. Details of CNN Architecture A based on Ref. [44].

Operation									Result		
Layer Number	**Layer Type**	**Filter Height**	**Filter Width**	**Filter Depth**	**Number of Filters**	**Padding**	**Stride**	**Total Parameters**	**Height**	**Width**	**Depth**
0	Input								100	100	3
1	2D Conv	25	25	3	24	Same	1	45,072	76	76	24
2	Max Pool	5	5	24			2		22	22	24
3	2D Conv	15	15	3	48	Same	1	32,544	11	11	48
4	Max Pool	2	2	48			2		1	1	48
5	2D Conv	11	11	3	96	Same	1	35,136	1	1	96
6	Flatten								1	1	96
7	Dropout (0.5)								1	1	96
8	ReLU								1	1	96
9	Dense				2			192	1	1	2
10	Softmax								1	1	2
						Total Parameters in Model		112,752			

Table 2. Details of CNN Architecture B, a hybrid model.

Operation									Result		
Layer Number	**Layer Type**	**Filter Height**	**Filter Width**	**Filter Depth**	**Number of Filters**	**Padding**	**Stride**	**Total Parameters**	**Height**	**Width**	**Depth**
0	Input								100	100	3
1	2D Conv	15	15	3	20	Same	1	13,560	100	100	20
2	ReLU								100	100	20
3	2D Conv	10	10	20	20	Valid	1	40,400	91	91	20
4	ReLU								91	91	20
5	Max Pool	5	5	20			5		18	18	20
6	Dropout (0.25)								18	18	20
7	2D Conv	5	5	20	40	Same	1	20,800	18	18	40
8	ReLU								18	18	40
9	2D Conv	2	2	40	40	Valid	1	8000	17	17	40
10	ReLU								17	17	40
11	Max Pool	4	4	40			4		4	4	40
12	Dropout (0.25)								4	4	40
13	Flatten								640	1	1
14	Dense				50			32,000	50	1	1
15	ReLU								50	1	1
16	Dropout (0.5)								50	1	1
17	Dense				2			100	2	1	1
18	Softmax								2	1	1
						Total Parameters in Model		114,860			

Two other architectures were also considered to improve results.

Another approach to modifying the architecture from Keras documentation was to scale everything up. Compared to the 32×32 pixel input of CIFAR, the 100×100 input needed for this model was almost exactly three times taller and wider. Therefore, all filter sizes were scaled up by three. The result has been named Architecture C and is shown in Table 3.

Comparing the three architectures, it is obvious that B and C are deeper (they have more layers). However, although Architecture B has more layers than Architecture A, it does not have many more parameters. This is because many of additional layers in B are activation layers and pooling layers which do not involve parameters. For this reason Architecture B was expected to do better than Architecture A. At the same time, Architecture C, though it does not have any filters as large as some in Architecture B, still has more than five times the number of parameters. The distribution of these parameters between layers is shown in Tables 1–3.

The RMSProp optimizer was used for training all three architectures. A learning rate of 0.001 was used, along with a decay of 1.0×10^{-6} [62], as implemented in the Keras

library [67]. The models were trained on batches of size 20, which were shuffled between epochs, for 500 epochs.

Table 3. Details of CNN Architecture C, inspired by the CIFAR example in Keras documentation.

		Operation							Result		
Layer Number	**Layer Type**	**Filter Height**	**Filter Width**	**Filter Depth**	**Number of Filters**	**Padding**	**Stride**	**Total Parameters**	**Height**	**Width**	**Depth**
0	Input								100	100	3
1	2D Conv	9	9	3	32	Same	1	7872	100	100	32
2	ReLU								100	100	32
3	2D Conv	9	9	32	32	Valid	1	83,968	92	92	32
4	ReLU								92	92	32
5	Max Pool	6	6	32			6		15	15	32
6	Dropout (0.25)								15	15	32
7	2D Conv	9	9	32	64	Same	1	167,936	15	15	64
8	ReLU								15	15	64
9	2D Conv	9	9	64	64	Valid	1	335,872	7	7	64
10	ReLU								7	7	64
11	Max Pool	6	6	64			6		1	1	64
12	Dropout (0.25)								1	1	64
13	Flatten								64	1	1
14	Dense				512			32,768	512	1	1
15	ReLU								512	1	1
16	Dropout (0.5)								512	1	1
17	Dense				2			1024	2	1	1
18	Softmax								2	1	1
						Total Parameters in Model		629,440			

2.5. Other Classifiers

To provide additional context for the performance of the deep CNNs, several simpler classifier models were considered. Without deep representation learning, these classifiers needed to be provided with appropriate features upon which to make the classification decision. In this case, a set of features was constructed based on expert knowledge, particularly that cracks induce dark shadows in the image patches. All image patches were first converted to greyscale. The first set of features were based on the premise that cracks appear at lower lightness than the brick face or mortar joints. Two thresholds T_1 and T_2 were selected manually based on our evaluation of the images. The first two features are simply the number of pixels below T_1 and between T_1 and T_2. However, it is not just the number of dark pixels but their arrangement that matters for identifying a crack. Therefore, the standard deviations of the coordinates of pixels within the selected thresholds (both x and y coordinates independently) were considered. Note that this may conflate small, disconnected regions of dark shadow with large single regions.

The thresholds described above define two sets of pixels:

$$S_1 : \text{Set of pixels with brightness below } T_1$$
$$S_2 : \text{Set of pixels with brightness between } T_1 \text{ and } T_2$$

This then leads to the definition of six input features for the classifiers:

$$X_1 = |S_1|$$
$$X_2 = |S_1|$$
$$X_3 = \sigma(S_1 x)$$
$$X_4 = \sigma(S_1 y)$$
$$X_5 = \sigma(S_2 x)$$
$$X_6 = \sigma(S_2 y)$$

where σ defines the standard deviation function and S_i indicates the labels of the set S_i.

These features were provided to a set of standard classification models: linear Support Vector Machine (SVM), Random Forest (RF), Gaussian Process (GP), Multi-Layer Perceptron (MLP), Naive Bayes (NB), and Quadratic Discriminant Analysis (QDA). The scikit-learn implementation was used for all the models [68]. For all models, the scikit-learn default parameters were used for simplicity; no hyperparameter tuning was performed. The GP model used a radial basis function kernel. Our use of these simple, "off the shelf" classifiers is intended to provide a baseline performance on our CNN models, rather than achieve optimal performance.

3. Results and Discussion

3.1. CNN Training

Training the models took around six minutes using four 2.60 GHz Intel Skylake cores and a single nVidia K20m GPU on a high-performance computing cloud. The accuracy of each model during training is shown in Figure 5. The lines show replicas of the same model training on a different data fold, initialized with different random initial weights, and under the effect of stochastic dropout. Interestingly, based on the training accuracy, Models A and B appear to learn at about the same rate and eventually reach about 95% accuracy, while Model C appears to learn more quickly and reach a higher accuracy of up to 99%. This is consistent with the greater trainable parameters in Model C, whereas Models A and B have about the same number.

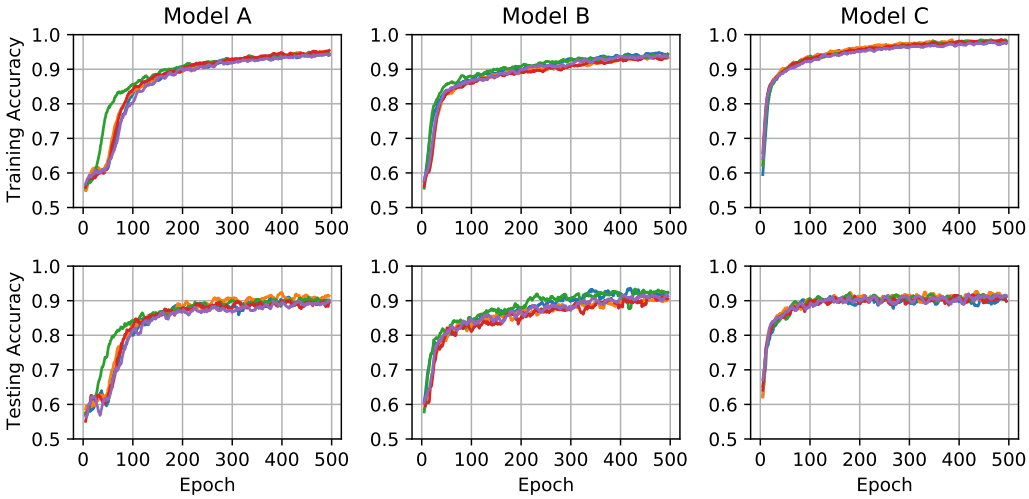

Figure 5. The accuracy of each architecture on training data (**top**) and testing data (**bottom**) during model training. Each line represents a different fold of the k-fold cross validation ($k = 5$) and are running averages over 10 epochs for clarity. Architectures correspond to those described in Tables 1–3.

However, as shown in the bottom row of Figure 5, the performance of each model on the testing data plateaued to much lower values than the training data, closer to 90% accuracy. The magnitude of this discrepancy reveals overfitting in cases where the model performed substantially better on training data compared to testing. This in turn indicates that the model will not be able to generalize to new images. Because the data set is small and there are so many trainable parameters in these models, overfitting is a major concern. From Figure 5, we observe that Models A and B had similar accuracy in training and testing and are therefore do not suffer significant overfitting. Model C, however, appears to have used its significantly greater number of parameters to overfit to the training data.

3.2. Testing on Laboratory Images

The "lab images" used for testing were from the same data set as the images used for training, but they were left out from training. Some standard metrics quantifying the performance of each model are shown in Table 4. Each result is the average of a 5-fold

cross-validation test, which was used to increase confidence in the results given such a small data set. The training and testing sets were always mutually exclusive. In each run, one subset (composed of 160 images) was reserved for testing, and the model was trained on the other four (totaling 650 images). Thus, each run represents a model trained on a slightly different data set and then tested on a different data set.

Table 4. Performance of three architectures tested on lab images not included in the training set. Reported values are the average and standard deviation over the five replicas shown in Figure 5. The best performer in each row is shown in bold.

	Model A	**Model B**	**Model C**
Accuracy	0.902 ± 0.014	**0.925** ± 0.011	0.887 ± 0.019
Precision	0.913 ± 0.035	**0.928** ± 0.021	0.882 ± 0.041
Recall	0.919 ± 0.042	**0.943** ± 0.008	0.930 ± 0.027
F1	0.915 ± 0.013	**0.936** ± 0.009	0.905 ± 0.014

Model B performed best in every metric shown in Table 4, followed closely by Model A; Model C performed the worst in three of the four metrics. Based on a one-sided t-test with $\alpha = 0.05$, Model B's performance was superior to Model A with statistical significance in accuracy and F1 score. Likewise, Model B's performance was superior to Model C in accuracy, precision, and F1 score. While the rest had $p > \alpha$, the metrics were still highest for Model B.

The confusion matrices for each model are shown in Figure 6. While the performance of each model varied, they showed similar qualitative behavior on the test data. In general, True positives were more likely than true negatives, and false positives were about twice as likely as false negatives, exposing a bias towards predicting cracks were present in the images. Interestingly, Model C's inferior performance appears to come almost exclusively from these false positives, with approximately 50% higher False Positive incidence compared to the other two architectures.

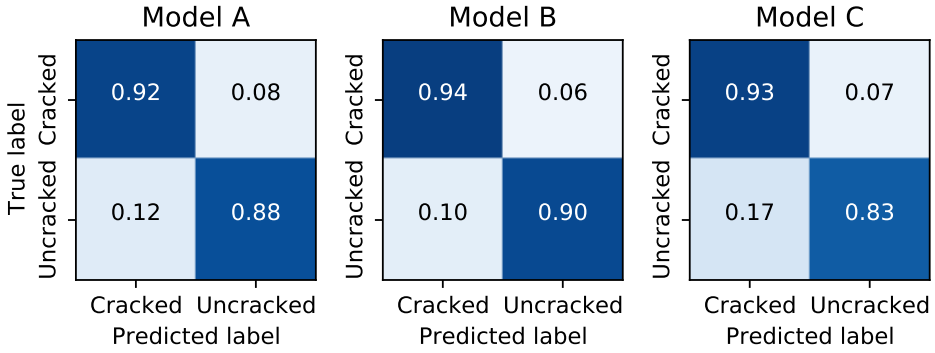

Figure 6. Confusion matrices for each architecture when tested on lab images not included in the training set.

We examine these failure modes of Model C in more detail through visual examples in Figure 7. It seems that the images that are easy to identify as cracked—such as those with deep, expansive cracks—are correctly labeled as such. Likewise, images with clean, ortho-rectified mortar joints are readily identified as uncracked. For the false negative case, there appear to be some uneven and messy mortar joints which confuse the classifer by obscuring small, hairline cracks. For the false positive case, several appear to also have uneven joints which might create extra shadows that mislead the classifier. The blue tint of some images does not appear to create a systematic problem.

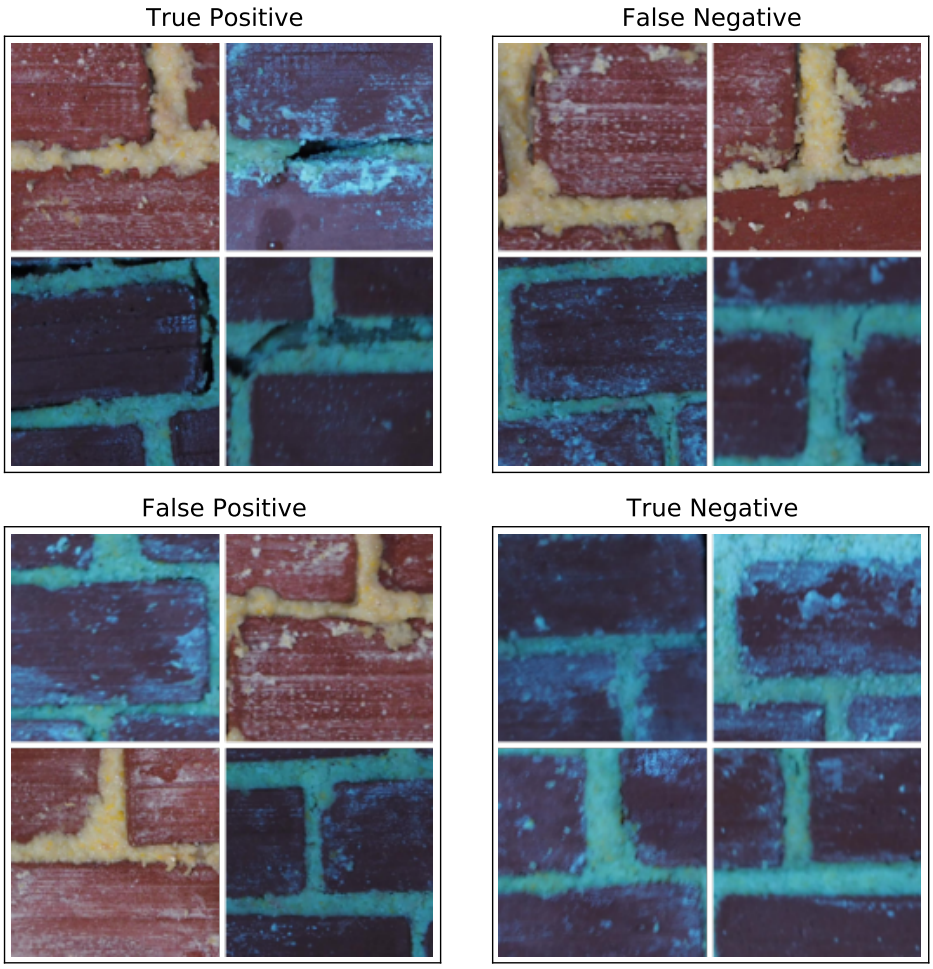

Figure 7. A random sample of four lab images from each case in the confusion matrix, as determined by Model C.

3.3. Domain Adaptation to Real-World Images

The relevant metrics when the CNNs are applied to these real-world images are presented in Table 5, and the associated confusion matrices in Figure 8. Interestingly, while the performance of all models was similar on the lab data (i.e., Figure 6), the model performance diverges significantly on the domain adaptation task. First, all models perform worse, which is to be expected since the real-world data set represents a much broader range of conditions. Somewhat surprisingly, Model A suffers about 50% greater deterioration in accuracy than Model B. This might be due to the shallower structure of the convolutional filters in Model A, which presumably are less capable of abstraction.

Table 5. Performance of three architectures tested on real-world images not included in the lab training set. Reported values are the average and standard deviation over the five replicas. The best performer in each row is shown in bold.

	Model A	**Model B**	**Model C**
Accuracy	0.615 ± 0.041	0.755 ± 0.012	**0.810** ± 0.051
Precision	0.650 ± 0.034	0.850 ± 0.010	**0.890** ± 0.046
Recall	0.607 ± 0.040	0.714 ± 0.034	**0.767** ± 0.043
F1	0.628 ± 0.020	0.776 ± 0.013	**0.824** ± 0.034

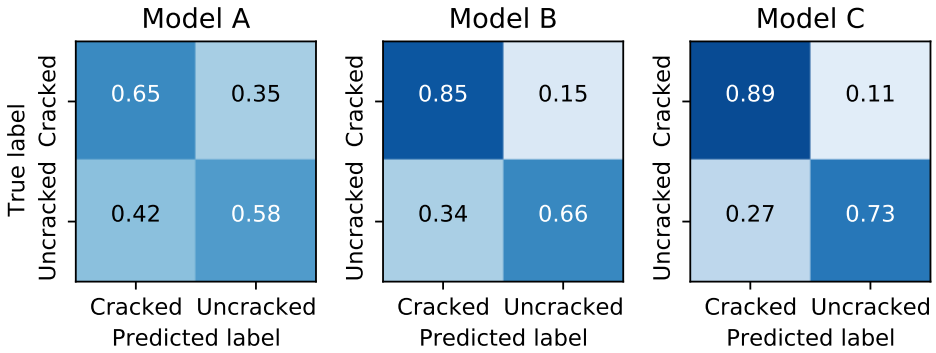

Figure 8. Confusion matrices for each architecture when tested on real-world images.

If these CNNs were to be deployed to study real structures, they would encounter a much greater variety of brick colors, sizes, textures, and other environmental conditions compared to those represented in the images of lab walls. We therefore studied the model performance on real-world images after being trained only on lab images, a process called domain adaptation. The models were tested against 90 different real-world images: 45 showing cracks and 45 with no visible cracks. Each of the walls had different relative sizes of bricks compared to images, different sizes of bricks compared to mortar joints, as well as non-uniform brick size in the same image (headers and stretchers). Furthermore, bricks and mortars of different colors were used of different colors were used, as well as different styles of mortar joints.

Equally surprising is that while Model C was substantially overfitted on the lab data, it suffers about 50% lower loss of accuracy compared to Model B, resulting in the best performance on the real-world images despite being the worst performer on the lab data. This suggests that the additional trainable parameters do provide an advantage in domain adaptation even though the performance on the lab data seemed to indicate this was not the case. We suspect this is a result of the lab images being very similar, creating an artificially smaller sample space compared to what the number of images would imply. Conversely, on the real-world images, a smaller number of images provides a large amount of variation. We hypothesize that the redundant filters learned by Model C are therefore able to provide additional information to the classifier, while the filters from Models A and B are less successful in adapting to the broader range of new images. Similarly, the deeper architecture of Model B compared to Model A appears more successful at generalizing from the lab to the real world.

We again examine these failure modes of Model C in more detail through visual examples in Figure 9. As in the lab data, the deepest cracks are easy to identify, as are uncracked images with clean mortar joints with faint or nonexistent shadows. For the false negative case, thin, straight cracks that are aligned with the mortar joints appear the hardest to detect. For the false positive case, it seems that strong shadows and discoloration on the brick face are the strongest contributors. We suspect that the color variation that occurs within a single real-world image is the primary reason for the high false positive rate compared to the lab images.

Figure 9. A random sample of four real-world images from each case in the confusion matrix, as determined by Model C.

3.4. Comparison to Other Classifiers

We also explored the use of simpler classifiers which are both less expensive to compute and more readily interpretable. As already described, the choice of classifiers was based on "off-the-shelf" models from the sci-kit learn Python package [68]. These models represent a variety of strategies including linear and non-linear methods, shallow neural networks and kernel methods, and ensemble methods.

Unsurprisingly, performance on the lab images was generally lower than the CNN models, as shown in Table 6. A notable exception is the high recall score of the MLP and QDA classifiers, which shows that those models throw very few false positives. For three of the four metrics, the highest performer was the RF model, implying that this might be the best alternative to the CNN.

Table 6. Performance of different ML models when trained on laboratory images and tested on unseen laboratory images. The best performer in each row is shown in bold.

	SVM	**RF**	**GP**	**MLP**	**NB**	**QDA**
Accuracy	0.836	**0.864**	0.841	0.841	0.822	0.822
Precision	0.755	**0.834**	0.761	0.777	0.712	0.719
Recall	0.912	0.846	0.912	0.879	**0.978**	0.956
F1	0.823	**0.842**	0.830	0.825	0.824	0.821

In addition to testing and training on lab images, we performed two additional cross-dataset validation studies. First, we trained on lab images and tested on real-world images, as we did with the CNNs. Second, we were also able to train the models on real-world images and then test on the lab images since these models have many fewer trainable

parameters and can be fit reasonably on our small real-world data set. The confusion matrices for each model under each of these three cases is shown in Figure 10.

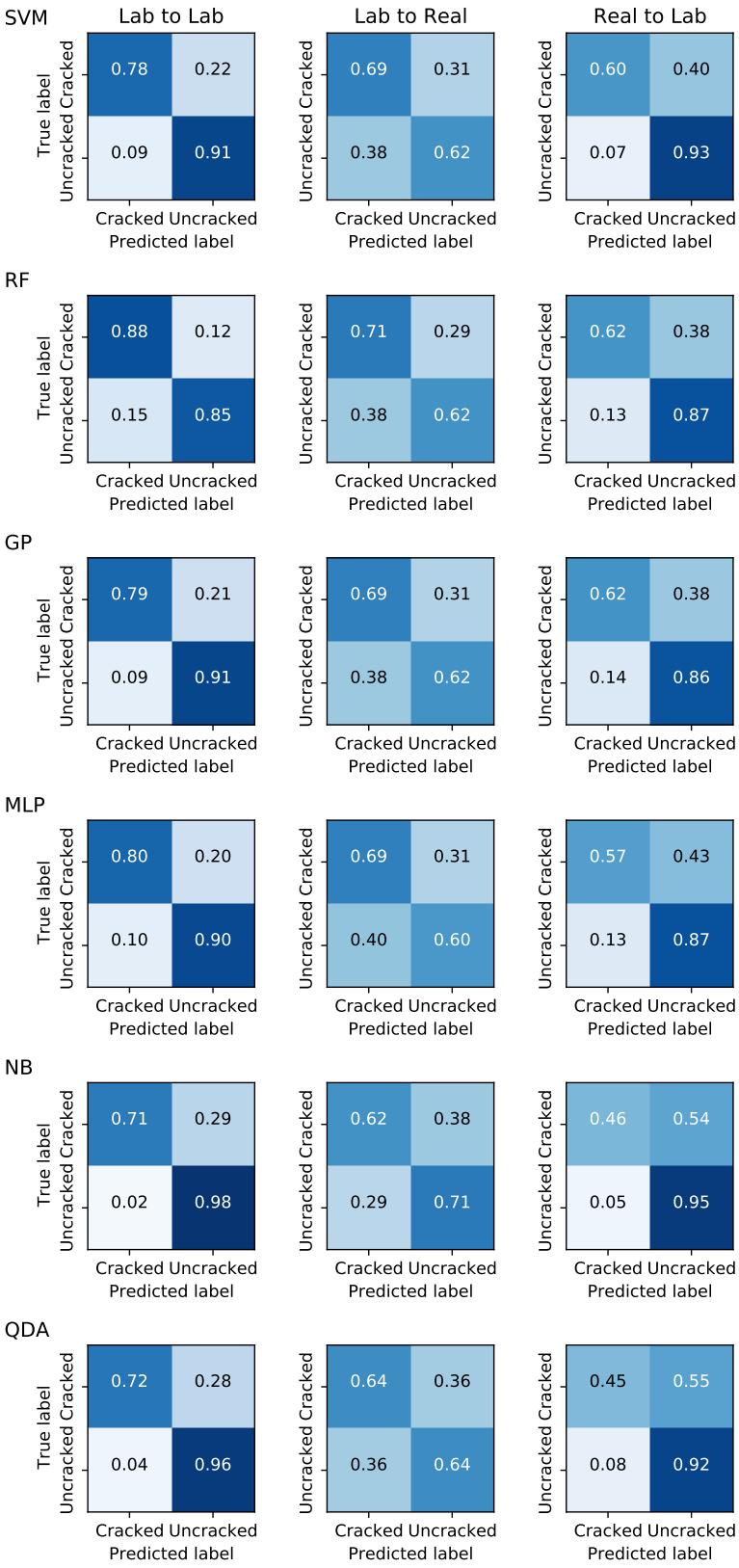

Figure 10. Confusion matrices demonstrating the performance of each classifier model (rows) under each train-test scenario (columns).

As expected, the performance drops significantly on the domain adaptation task (Lab to Real). The models appear to segregate into two categories, with SVM, RF, GP, and MLP performing roughly equally, and NB and QDA performing notably worse. The advantage of RF over the other three in the former category appears to dissipate for domain adaptation, with only 2% better true positive rate. This implies overfitting by the RF model in the Lab to Lab scenario. Overall, these models appear comparable, although we note that SVM and RF train significantly faster than GP and MLP.

On the reverse domain adaptation task (Real to Lab), we again find similar performance by models in each category, with the top four performing stronger than the bottom two. Notably, the false positive rate is much lower when training on real-world images and testing on lab images compared to the reverse case, while false negatives are higher. This implies that the variety of cracks is greater in the real-world data, such that the models learn the general features of a crack, whereas in the lab data there are subtle cracks which are more difficult to detect without training on them, especially in the presence of uncracked images with strong shadows and discoloration as shown in Figure 9.

An advantage of these simple models (i.e., models without built-in representation learning) is that we can evaluate how strongly they rely on each provided feature. Since the SVM and MLP had the highest accuracy in the Real to Lab case, we also performed a permutation feature importance test using those classifiers. For each feature, the model was refit with that feature shuffled, such that it no longer corresponded to each other feature in the feature vector. Then the model accuracy was compared to the baseline to quantify the amount of information in that feature.

This test was performed for both the lab and real-world data sets, with results shown in Figures 11 and 12, respectively. Comparing the results, we see that in the lab case both models relied mainly on Feature 5, the standard deviation of x-coordinates of S_2. SVM relies more heavily on the pixels in S_1 while MLP uses S_2, while the remaining features are less significant. The positive values of the MLP probably indicate that the model is not fully converged (we used the default settings with no hyperparameter tuning).

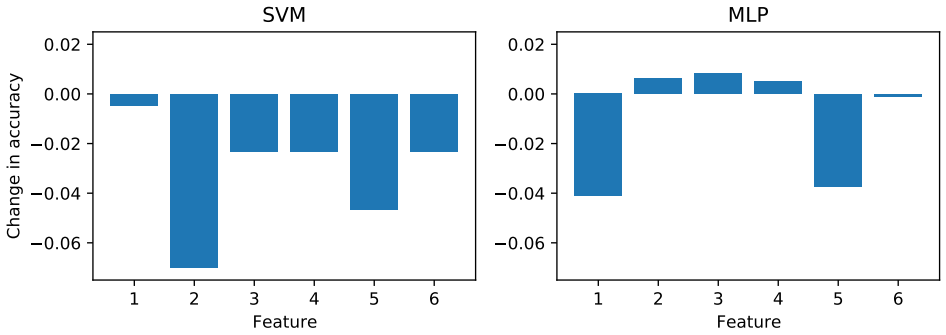

Figure 11. Permutation feature importance for the SVM and MLP models when trained and tested on the laboratory images.

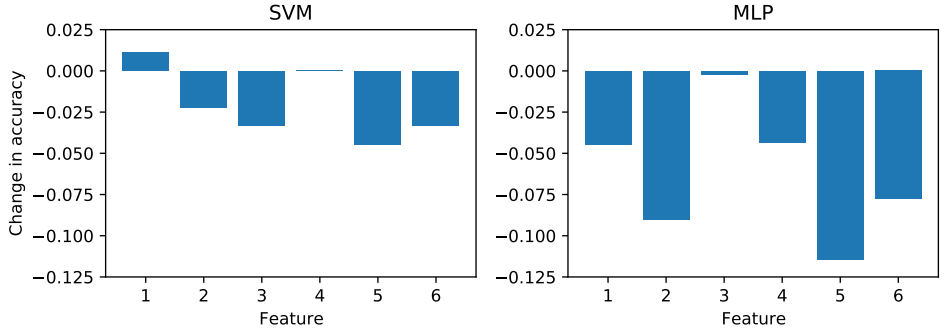

Figure 12. Permutation feature importance for the SVM and MLP models when trained and tested on the real-world images.

When considering the real-world case, both models use a completely different scheme, although Feature 5 remains prominent. The fact that the x-coordinate (Feature 5) is used more heavily than the y-coordinate (Feature 6) might reveal a bias in the data set towards cracks of a certain orientation. Likewise, the dominant use of only two features by the MLP in the lab training case is evidence of overfitting to a particular aspect of those data.

4. Conclusions

In this work, we produced a new dataset consisting of 2542 labeled image patches of masonry walls in a controlled laboratory environment. These data were used to train three different CNN architectures to classify image patches as cracked or uncracked, a challenging problem which has been the subject of several recent studies by other authors. The results show that the same CNN architecture which was sufficient for concrete and asphalt cracking in Ref. [36] is not sufficient for crack detection in masonry structures. Here, we were able to overcome the limitation of fixed-size sliding windows which have been necessary in past studies to handle the heterogeneity of masonry. We also showed that deeper networks provide superior results on test data.

Additionally, we have demonstrated the ability of the proposed architecture to perform well in domain adaptation to images found on the internet with different colors and relative sizes of materials compared to the training data. While we observed overfitting on the laboratory data for the CNN model with the most trainable parameters, these extra parameters provided superior performance on the domain adaptation task. The best model gave an accuracy of 81.0% under these circumstances, while the model which performed best on the lab images gave only 61.5% accuracy. This contrasts strongly with the 88.7% to 92.5% accuracy achieved on the lab images, and indicates that good model performance on curated, homogeneous data does not necessarily translate to real-world conditions.

The CNN model performance was also compared to that of six simple classifiers based on handcrafted features. The features were constructed from greyscale image patches which focused on dark regions indicative of cracking (i.e., deep shadows). Our results showed that four of the models, the SVM, RF, GP, and MLP, performed better than the remaining two, NB and QDA. While faster to train and run, these simple classifiers performed worse on the lab testing set as well as on the real-world testing set. However, we found that training these on real-world images provided superior domain adaptation performance when going in reverse, from only 90 real-world images to hundreds of lab images. We hypothesize that the greater variety in the real-world data produces models which are able to capture the narrow range of conditions in the lab as a subset, while the reverse results in the inability to generalize.

We conclude that successful domain adaptation is possible in both the CNN and simpler classifiers if trained on a wide range of masonry shapes, colors, and lighting conditions, and will lead to more accurate models than training on a more homogeneous data set (e.g., our controlled lab images). Some engineering firms already have large sets of facade images which they could leverage for this purpose. One outstanding question is that of visual clutter in the image patches, such as doors, windows, lights, and other background. While outside the scope of this work, future crack classification models meant for deployment in structural health monitoring contexts should consider how to separate such background clutter from structurally relevant foreground.

Author Contributions: Conceptualization, M.J.H., R.K.N. and B.G.; methodology, M.J.H., R.K.N. and W.F.R.; laboratory experiments, M.J.H.; model design and implementation M.J.H., R.K.N. and W.F.R.; writing—original draft preparation, M.J.H. and R.K.N.; writing—review and editing, W.F.R. and B.G. All authors have read and agreed to the published version of the manuscript.

Funding: This paper is based on work in part supported by the National Science Foundation Graduate Research Fellowship Program under Grant no. DGE-1656466. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the National Science Foundation. Additional supported was provided by the Dean's Fund for Innovation at Princeton and the Department of Civil and Environmental Engineering.

Data Availability Statement: The data presented in this study are openly available in Zenodo at https://doi.org/10.5281/zenodo.5108846 (accessed on 1 June 2021).

Acknowledgments: We thank Peter Chen, Michael Guerzhoy, Byung Kwan Oh, David Luet, Daniel Clark, Daniel Cook, Warren Powell, and Jianqing Fan for fruitful discussions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rippmann, M.; Block, P. Rethinking structural masonry: Unreinforced, stone-cut shells. *Proc. Inst. Civ. Eng. Constr. Mater.* **2013**, *166*, 378–389. [CrossRef]
2. Kamal, O.; Hamdy, G.; El-Salakawy, T. Nonlinear analysis of historic and contemporary vaulted masonry assemblages. *HBRC J.* **2014**, *10*, 235–246. [CrossRef]
3. Mısırlısoy, D. Analysis of the Structure and Design Relationship between Contemporary Extensions and Remodeled Masonry Buildings. Ph.D. Thesis, Eastern Mediterranean University (EMU), Gazimağusa, Turkey, 2011.
4. Sun, M.T.; Ochsendorf, J.A. Nervi's Design and Construction Methods of Two Thin-Shell Structures: The Leverone Field House and Thompson Arena. In Proceedings of the IASS Annual Symposia, Boston, MA, USA, 16–20 July 2018; Volume 2018, pp. 1–8.
5. Block, P.; Van Mele, T.; Liew, A.; DeJong, M.; Escobedo, D.; Ochsendorf, J.A. Structural design, fabrication and construction of the Armadillo vault. *Struct. Eng. J. Inst. Struct. Eng.* **2018**, *96*, 10–20.
6. Porst, C.; Brzev, S.; Ochsendorf, J. Confined Masonry for Resilient Low-Cost Housing in India: A Design and Analysis Method. In Proceedings of the 16th World Conference on Earthquake Engineering 2017, Santiago, Chile, 9–13 January 2017.
7. Napolitano, R.; Douglas, I.; Garlock, M.; Glisic, B. Virtual tour environment of Cuba's National School of Art. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci* **2017**, *42*, W5. [CrossRef]
8. Douglas, I.; Napolitano, R.; Garlock, M.; Glisic, B. Reconsidering the vaulted forms of cuba's national school of ballet. In *Structural Analysis of Historical Constructions*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 2150–2158.
9. Ritchie, T.; Davison, J. Moisture content and freeze-thaw cycles of masonry materials. *J. Mater.* **1968**, *3*, 658–671.
10. Scherer, G.W. Internal stress and cracking in stone and masonry. In *Measuring, Monitoring and Modeling Concrete Properties*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 633–641.
11. Uranjek, M.; Bokan-Bosiljkov, V. Influence of freeze–thaw cycles on mechanical properties of historical brick masonry. *Constr. Build. Mater.* **2015**, *84*, 416–428. [CrossRef]
12. Leeanansaksiri, A.; Panyakapo, P.; Ruangrassamee, A. Seismic capacity of masonry infilled RC frame strengthening with expanded metal ferrocement. *Eng. Struct.* **2018**, *159*, 110–127. [CrossRef]
13. Grimmer, A.E. *A Glossary of Historic Masonry Deterioration Problems and Preservation Treatments*; Department of the Interior, National Park Service, Preservation Assistance: Washington, DC, USA, 1984.
14. Scherer, G.W.; Flatt, R.; Wheeler, G. Materials science research for the conservation of sculpture and monuments. *MRS Bull.* **2001**, *26*, 44–50. [CrossRef]
15. Burd, H.; Houlsby, G.; Augarde, C.; Liu, G. Modelling tunnelling-induced settlement of masonry buildings. *Proc. Inst. Civ. Eng. Geotech. Eng.* **2000**, *143*, 17–29. [CrossRef]
16. Giardina, G.; Hendriks, M.A.; Rots, J.G. Assessment of the settlement vulnerability of masonry buildings. In Proceedings of the 1st WTA International PhD Symposium-Buildings Materials and Building Technology for Preservation of the Built Heritage, Leuven, Belgium, 8–9 October 2009.
17. Giardina, G.; Van de Graaf, A.V.; Hendriks, M.A.; Rots, J.G.; Marini, A. Numerical analysis of a masonry façade subject to tunnelling-induced settlements. *Eng. Struct.* **2013**, *54*, 234–247. [CrossRef]
18. Block, P.; DeJong, M.; Davis, L.; Ochsendorf, J. Tile vaulted systems for low-cost construction in Africa. *ATDF J.* **2010**, *7*, 4–13.
19. Ingham, J.; Griffith, M. Performance of unreinforced masonry buildings during the 2010 Darfield (Christchurch, NZ) earthquake. *Aust. J. Struct. Eng.* **2010**, *11*, 207–224. [CrossRef]
20. Federal Emergency Management Agency. Unreinforced Masonry Buildings and Earthquakes. Developing Successful Risk Reduction Programs. FEMA P-774. Buildings. 2009, 53p. Available online: https://www.madcad.com/media/fema/FEMA-P774-2009.pdf (accessed on 1 June 2021).
21. Orbán, Z. UIC project on assessment, inspection and maintenance of masonry arch railway bridges. *ARCH* **2007**, *7*, 3–12.
22. Gentile, C.; Saisi, A. On-site investigation and dynamic monitoring for the post-earthquake assessment of a masonry tower. In Proceedings of the 9th International Conference on Structural Analysis of Historical Constructions, Mexico City, Mexico, 14–17 October 2014.
23. Ellenberg, A.; Kontsos, A.; Bartoli, I.; Pradhan, A. Masonry crack detection application of an unmanned aerial vehicle. In Proceedings of the Computing in Civil and Building Engineering (2014), Orlando, FL, USA, 23–25 June 2014; ASCE: Reston, VA, USA, 2014; pp. 1788–1795.
24. Blyth, A.; Napolitano, R.; Glisic, B. Documentation, structural health monitoring and numerical modelling for damage assessment of the Morris Island Lighthouse. *Philos. Trans. R. Soc. A* **2019**, *377*, 20190002. [CrossRef] [PubMed]
25. Kunal, K.; Killemsetty, N. Study on control of cracks in a Structure through Visual Identification & Inspection. *IOSR J. Mech. Civ. Eng.* **2014**, *11*, 64–72.

26. Torok, M.M.; Golparvar-Fard, M.; Kochersberger, K.B. Image-Based Automated 3D Crack Detection for Post-disaster Building Assessment. *J. Comput. Civ. Eng.* **2014**, *28*, A4014004. [CrossRef]
27. Yeum, C.M.; Dyke, S.J.; Ramirez, J. Visual data classification in post-event building reconnaissance. *Eng. Struct.* **2018**, *155*, 16–24. [CrossRef]
28. Groeger, J.L.; Stephanos, P.; Dorsey, P.; Chapman, M. Implementation of automated network-level crack detection processes in Maryland. *Transp. Res. Rec.* **2003**, *1860*, 109–116. [CrossRef]
29. Cord, A.; Chambon, S. Automatic road defect detection by textural pattern recognition based on AdaBoost. *Comput. Aided Civ. Infrastruct. Eng.* **2012**, *27*, 244–259. [CrossRef]
30. Koch, C.; Jog, G.M.; Brilakis, I. Automated Pothole Distress Assessment Using Asphalt Pavement Video Data. *J. Comput. Civ. Eng.* **2013**, *27*, 370–378. [CrossRef]
31. Zalama, E.; Gómez-García-Bermejo, J.; Medina, R.; Llamas, J. Road crack detection using visual features extracted by Gabor filters. *Comput. Aided Civ. Infrastruct. Eng.* **2014**, *29*, 342–358. [CrossRef]
32. Gopalakrishnan, K.; Smadi, O.G.; Ceylan, H.; Celik, K.; Somani, A.K. *Machine-Vision-Based Roadway Health Monitoring and Assessment: Development of a Shape-Based Pavement-Crack-Detection Approach*; Iowa State University: Ames, IA, USA, 2016.
33. Zhang, L.; Yang, F.; Zhang, Y.D.; Zhu, Y.J. Road crack detection using deep convolutional neural network. In Proceedings of the IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; pp. 3708–3712.
34. Zhang, A.; Wang, K.C.; Li, B.; Yang, E.; Dai, X.; Peng, Y.; Fei, Y.; Liu, Y.; Li, J.Q.; Chen, C. Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network. *Comput. Aided Civ. Infrastruct. Eng.* **2017**, *32*, 805–819. [CrossRef]
35. Yeum, C.M.; Dyke, S.J. Vision-Based Automated Crack Detection for Bridge Inspection. *Comput. Aided Civ. Infrastruct. Eng.* **2015**, *30*, 759–770. [CrossRef]
36. Cha, Y.J.; Choi, W.; Suh, G.; Mahmoudkhani, S.; Büyüköztürk, O. Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. *Comput. Aided Civ. Infrastruct. Eng.* **2018**, *33*, 731–747. [CrossRef]
37. Xu, Y.; Bao, Y.; Chen, J.; Zuo, W.; Li, H. Surface fatigue crack identification in steel box girder of bridges by a deep fusion convolutional neural network based on consumer-grade camera images. *Struct. Health Monit.* **2019**, *18*, 653–674. [CrossRef]
38. Dung, C.V.; Sekiya, H.; Hirano, S.; Okatani, T.; Miki, C. A vision-based method for crack detection in gusset plate welded joints of steel bridges using deep convolutional neural networks. *Autom. Constr.* **2019**, *102*, 217–229. [CrossRef]
39. Abdel-Qader, I.; Abudayyeh, O.; Kelly, M.E. Analysis of edge-detection techniques for crack identification in bridges. *J. Comput. Civ. Eng.* **2003**, *17*, 255–263. [CrossRef]
40. Nishikawa, T.; Yoshida, J.; Sugiyama, T.; Fujino, Y. Concrete crack detection by multiple sequential image filtering. *Comput. Aided Civ. Infrastruct. Eng.* **2012**, *27*, 29–47. [CrossRef]
41. Hu, D.; Tian, T.; Yang, H.; Xu, S.; Wang, X. Wall crack detection based on image processing. In Proceedings of the Third International Conference on Intelligent Control and Information Processing, Dalian, China, 15–17 July 2012; pp. 597–600.
42. Amer, G.M.H.; Abushaala, A.M. Edge detection methods. In Proceedings of the 2nd World Symposium on Web Applications and Networking (WSWAN), Sousse, Tunisia, 21–23 March 2015; pp. 1–7.
43. Wang, N.; Zhao, Q.; Li, S.; Zhao, X.; Zhao, P. Damage classification for masonry historic structures using convolutional neural networks based on still images. *Comput. Aided Civ. Infrastruct. Eng.* **2018**, *33*, 1073–1089. [CrossRef]
44. Cha, Y.J.; Choi, W.; Büyüköztürk, O. Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks. *Comput. Aided Civ. Infrastruct. Eng.* **2017**, *32*, 361–378. [CrossRef]
45. Ali, L.; Alnajjar, F.; Jassmi, H.A.; Gochoo, M.; Khan, W.; Serhani, M.A. Performance Evaluation of Deep CNN-Based Crack Detection and Localization Techniques for Concrete Structures. *Sensors* **2021**, *21*, 1688. [CrossRef]
46. Kim, B.; Yuvaraj, N.; Preethaa, K.S.; Pandian, R.A. Surface crack detection using deep learning with shallow CNN architecture for enhanced computation. *Neural Comput. Appl.* **2021**, *33*, 9289–9305. [CrossRef]
47. Reghukumar, A.; Anbarasi, L.J. Crack Detection in Concrete Structures Using Image Processing and Deep Learning. In *Advances in Electrical and Computer Technologies: Select Proceedings of ICAECT 2020*; Springer: Singapore, 2021; pp. 211–219.
48. Chehri, A.; Saeidi, A. IoT and Deep Learning Solutions for an Automated Crack Detection for the Inspection of Concrete Bridge Structures. In *International Conference on Human-Centered Intelligent Systems, Proceedings of the KES-HCIS 2021: Human Centred Intelligent Systems, Virtual Conference, 14–16 June 2021*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 110–119.
49. Bang, H.; Min, J.; Jeon, H. Deep Learning-Based Concrete Surface Damage Monitoring Method Using Structured Lights and Depth Camera. *Sensors* **2021**, *21*, 2759. [CrossRef] [PubMed]
50. Rezaie, A.; Achanta, R.; Godio, M.; Beyer, K. Comparison of crack segmentation using digital image correlation measurements and deep learning. *Constr. Build. Mater.* **2020**, *261*, 120474. [CrossRef]
51. Chow, J.K.; Su, Z.; Wu, J.; Tan, P.S.; Mao, X.; Wang, Y.H. Anomaly detection of defects on concrete structures with the convolutional autoencoder. *Adv. Eng. Inform.* **2020**, *45*, 101105. [CrossRef]
52. Lappas, D.; Argyriou, V.; Makris, D. Fourier transformation autoencoders for anomaly detection. In Proceedings of the ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 6–11 June 2021; pp. 1475–1479.
53. Pena, L.; Domede, N.; Fady, N. Contribution of PCA and ANN in the Structural Diagnosis of a Masonry Lighthouse under Temperature and Wind Actions. *Int. J. Archit. Herit.* **2021**, 1–22. [CrossRef]
54. Xu, J.; Yu, X. Detection of concrete structural defects using impact echo based on deep networks. *J. Test. Eval.* **2020**, *49*, 109–120.

55. Pyle, R.J.; Bevan, R.L.; Hughes, R.R.; Rachev, R.K.; Ali, A.A.S.; Wilcox, P.D. Deep Learning for Ultrasonic Crack Characterization in NDE. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control* **2021**, *68*, 1854–1865. [CrossRef]
56. Medak, D.; Posilović, L.; Subašić, M.; Budimir, M.; Lončarić, S. Automated Defect Detection from Ultrasonic Images Using Deep Learning. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control* **2021**. [CrossRef]
57. Noori Hoshyar, A.; Kharkovsky, S.N.; Samali, B. Statistical features and traditional SA-SVM classification algorithm for crack detection. *J. Signal Inf. Process.* **2018**, *9*, 111–121.
58. Chaiyasarn, K.; Sharma, M.; Ali, L.; Khan, W.; Poovarodom, N. Crack detection in historical structures based on convolutional neural network. *Int. J. Geomate* **2018**, *15*, 240–251. [CrossRef]
59. Chaiyasarn, K.; Khan, W.; Ali, L.; Sharma, M.; Brackenbury, D.; DeJong, M. Crack Detection in Masonry Structures using Convolutional Neural Networks and Support Vector Machines. In Proceedings of the International Symposium on Automation and Robotics in Construction, Berlin, Germany, 20–25 July 2018; Volume 35, pp. 1–8.
60. Luqman, A.; Khanand, W.; Chaiyasarn, K. Damage detection and localization in masonry structure using faster region convolutional networks. *Int. J. Geomate* **2019**, *17*, 98–105.
61. Wang, N.; Zhao, X.; Zhao, P.; Zhang, Y.; Zou, Z.; Ou, J. Automatic damage detection of historic masonry buildings based on mobile deep learning. *Autom. Constr.* **2019**, *103*, 53–66. [CrossRef]
62. Agyemang, D.B.; Bader, M. Surface Crack Detection Using Hierarchal Convolutional Neural Network. In *UK Workshop on Computational Intelligence, Proceedings of the UKCI 2019: Advances in Computational Intelligence Systems, Portsmouth, UK, 4–6 September 2019*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 173–186.
63. Napolitano, R.K.; Glisic, B. Minimizing the adverse effects of bias and low repeatability precision in photogrammetry software through statistical analysis. *J. Cult. Herit.* **2018**, *31*, 46–52. [CrossRef]
64. Napolitano, R.; Hess, M.; Glisic, B. Integrating non-destructive testing, laser scanning, and numerical modeling for damage assessment: The room of the elements. *Heritage* **2019**, *2*, 151–168. [CrossRef]
65. Napolitano, R.; Glisic, B. Methodology for diagnosing crack patterns in masonry structures using photogrammetry and distinct element modeling. *Eng. Struct.* **2019**, *181*, 519–528. [CrossRef]
66. Napolitano, R.K.; Hess, M.; Glisic, B. The Foundation Walls of the Baptistery Di San Giovanni: A Combination of Laser Scanning and Finite-Distinct Element Modeling to Ascertain Damage Origins. *Int. J. Archit. Herit.* **2019**, 1–14. [CrossRef]
67. Keras: The Python Deep Learning Library. Documentation. 2019. Available online: https://ui.adsabs.harvard.edu/abs/2018ascl.soft06022C/abstract (accessed on 1 June 2021)
68. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.