Engineering a Multi-Label Classifier for Damaged Water Infrastructure Systems

Ben Saalberg
*Kalamazoo College, Kalamazoo, MI 49006**
(Dated: December 2, 2024)

Machine learning techniques have gained significant interest in recent years for their wide-ranging applications in engineering fields. One lesser-explored opportunity for civil engineers is damage or defect classification. Currently, if a system fails an expert is required to diagnose the issue and recommend a solution. Machine learning models have the potential to bypass this process by directly classifying the defect without the need for a human perspective. In our research, we investigate the effectiveness of a variety of simple and advanced classification models in identifying potential defects in water infrastructure systems. We find that simple models such as decision trees and logistic regression are insufficient to properly classify data of such high complexity. Artificial neural networks fare slightly better, while boosted convolutional networks outperform expectations and achieve perfect accuracy on the test data.

I. BACKGROUND

Machine learning techniques have already demonstrated that computer programs can replace humans in a variety of day-to-day tasks. One particularly promising area is diagnosis and classification. In the field of engineering, quality assurance is an essential role that has traditionally been reserved for experts who have a detailed understanding of a given system. These engineers are able to diagnose potential problems and recommend the proper course of action, but the process can be costly both in terms of time and money. Delays due to personnel or monetary shortages can even become deadly in emergency situations.

Image recognition and classification programs have the exciting potential to streamline this process significantly while benefiting all parties involved. If damage could be assessed as easily as taking a picture of the piece of infrastructure, the owners would save money on inspection costs, the users and public would be safer, and engineers would be free to develop new technologies rather than being called upon to maintain older systems.

In our research, we investigate the effectiveness of a variety of classification models for identifying damage in water infrastructure. This application is relatively unexplored, largely due to the complexity of the problem and the similarities between different types of damage. For example, the difference between a crack and a rupture can be nuanced. This level of detail can be difficult to program, but a successful classifier can be built with proper model selection and management of hyperparameters. While simple models may still struggle with such complex data, we intend to test those models first before moving on to more advanced models such as neural networks.

* jonah.beurkens20@kzoo.edu; benjamin.saalberg21@kzoo.edu

II. DATA PREPARATION

The foundation of any successful ML project lies in the quality and preparation of its data. For this model, the data preparation process involved several key steps. First and foremost was to compile a comprehensive dataset. Our dataset had 80 total images, with the following augmentations: [1]

Augmentation	Type
Flip	Horizontal, Vertical
90° Rotation	Clockwise, Counterclockwise
Small-Angle Rotation	$-15°$ to $15°$
Shear	$\pm 15°$ Horizontal, $\pm 18°$ Vertical
Blur	Up to 2 pixels
Noise	Up to 4% of pixels
Mosaic	4 Images

We split the dataset into a training set (68 images, 85%), test set (5 images, 6%), and validation set (7 images, 9%).

Our initial plan for preprocessing, given the high-dimensional nature of the raw images, involved simplifying the data without losing critical information. This included converting all images to grayscale to reduce the data's dimensionality while preserving essential structural features necessary for identifying conditions like cracks and holes. Finally, the pre-processed images underwent feature extraction, where each image's pixel values were flattened into a single array. This transforms the image data into a format suitable for the ML model to capture the spatial hierarchy of the pixels which in turn convey the condition of the pipes. Each image was scaled to $100 \cdot 100$ pixels to reduce the computational resources required. At this size, each image has 10,000 features - one for each pixel. Each image is associated with a binary matrix representing its class(es). These

labels have three classes: **crack, hole,** and **rupture.** The binary matrix label contains a 1 in each position that corresponds to a present defect and a 0 in each position that corresponds to an absent defect. Each image can belong to any number of classes, including none of them.

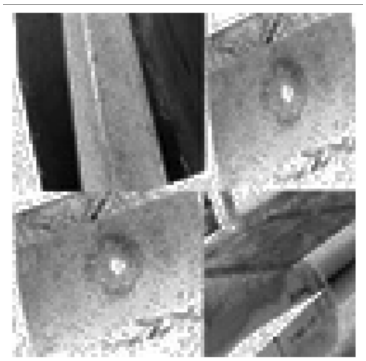

FIG. 1. A sample image from the training set after the grayscale pre-processing was completed.

After our initial experimentation, we identified the data preparation process as an area of potential improvement. We found that our first models performed poorly on the grayscale dataset, and decided to process the images differently in an effort to preserve more of the original image integrity. The most important change we made was to keep the images in their original colors. The revised preprocessing plan was very similar to our initial attempt described above, except the RGB values of each pixel were recorded rather than the grayscale values. We kept the images at a size of $100 \cdot 100$ pixels, resulting in 30,000 features per image: three color values per pixel, with 10,000 pixels. Figures 1 and 2 show samples of preprocessed images using the grayscale and color methods, respectively.

FIG. 2. Another sample image after the color pre-processing was completed. Features are more clearly visible.

III. PRELIMINARY METHODS AND RESULTS

A. Logistic Regression

First, we implemented a Logistic Regression model encapsulated within a "MultiOutputClassifier". This approach was predicated on the model's efficacy in binary classification and its simplicity. The model was fit to the entire training set with no regularization, since all features represent the same thing and should therefore be treated equally. However, the logistic regression model's performance, as determined through cross-validation techniques, was sub-optimal. We conducted a 5-fold cross-validation accuracy test with the following results:

Fold	Accuracy Score
1	0.21
2	0.43
3	0.43
4	0.38
5	0.23

B. Decision Tree

We then directed our attention towards a Decision Tree Classifier. This model was selected for its enhanced flexibility in capturing non-linear relationships between the features and the target variables, an attribute not as prominently featured in logistic regression models. We trained the model with default hyperparameters, and assessed its performance using 5-fold cross-validation:

Fold	Accuracy Score
1	0.36
2	0.29
3	0.43
4	0.31
5	0.23

C. Random Forest

In an attempt to use a more robust framework we deployed a Random Forest Classifier. This model is recognized for its proficiency in handling high-dimensional data and reduced propensity for overfitting through ensemble learning. We trained the model using default hyperparameters, and again used 5-fold cross-validation to assess its accuracy:

Fold	Accuracy Score
1	0.21
2	0.43
3	0.43
4	0.46
5	0.38

IV. ADVANCED METHODS AND RESULTS

A. Artificial Neural Network

Following the poor performance of the preliminary models, we turned to neural networks for their increased ability to handle complex data. We started by training an artificial neural network (ANN) using Tensorflow's Keras packages.

An important additional preprocessing step was performed before training the neural network models: all features were normalized by dividing the RGB values by their maximums.

We built an ANN with four hidden layers:

```
model = keras.Sequential([
    layers.Dense(256, activation='relu',
                 input_shape=(100 * 100 * 3,)),
    layers.Dropout(0.5),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(num_classes,
                 activation='softmax')
])
```

The model consisted of alternating dense and dropout layers. The dense layers had 256 and 128 neurons respectively, and each dropout layer removed half of the features to avoid overfitting. Both dense layers used ReLU activation functions, while the output layer used the softmax activation function to determine which classes an image belonged to. The model was compiled using the Adam optimizer with a learning rate of 0.001, categorical crossentropy as the loss function, and accuracy as the evaluation metric. It was trained with a batch size of 32 over 10 epochs.

We recorded a 60% accuracy on the test data for the artificial neural network.

B. Convolutional Neural Network

Finally, we engineered a more complex convolutional neural network (CNN) model. To further amplify the model's learning capacity, we utilized AdaBoosting with 5 iterations. Each CNN was built with the following specifications:

```
model = keras.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu',
                  input_shape=(100, 100, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(3, activation='softmax')
])
```

In this configuration, each of the estimators has 7 hidden layers (three convolutional layers, two max pooling layers, a flatten layer, and a dense layer). The convolutional layers had 32, 64, and 64 neurons respectively, and all three used a $3 \cdot 3$ kernel with the ReLU activation function. The max pooling layers used a kernel size of $2 \cdot 2$. The flatten layer was necessary to transform the shape of the input to the dense layer, which needs a 1-dimensional input. The dense layer had 64 neurons, also with the ReLU activation function. The output layer contained 3 neurons, using the softmax activation function.

As before, the models were compiled using the Adam optimizer (learning rate 0.001), categorical crossentropy loss function, and accuracy as the evaluation metric. The models were trained with a batch size of 30,000 over 10 epochs.

We recorded an accuracy of 100% on the test data using our AdaBoosted CNN model.

V. DISCUSSION

Our model development process was driven largely by the results we achieved at each step along the way. We began by training the simplest classification models, so we were not surprised to record poor results. In fact, none of our preliminary models achieved an accuracy score of 50% in any validation fold. However, we expected that this would be the case because the data is significantly more complicated than other projects we had already completed. These basic models work well for well-defined images such as the numbers MNIST dataset that we had practiced on in the past, but not so well for images where the important features are more varied and also obscured by unimportant aspects. For this reason, we conducted minimal experimentation with any hyperparameters; we were satisfied with checking on the performance of the models with default hyperparameters, but quickly moved to the advanced models. It is highly likely that hyperparameter optimization would

enhance the results of all three preliminary models, but we anticipated that the neural networks would outperform even the optimized basic models.

That hypothesis was largely proven correct, with the ANN model reaching 60% accuracy on the test data and the AdaBoosted CNN reaching perfect 100% accuracy. Notably, neither model was optimized due to computational limitations. All of our work was done in Google Colab, which offers only a small amount of RAM to users on the free plan. We had already reached the maximum allotted RAM just by training the models on higher-resolution images, so there was no room to perform a grid search for the optimal hyperparameters or test various models with more hidden layers. We think this is especially important for the performance of the ANN model, which had a significantly lower accuracy than the CNN model. Had we been able to compute the best hyperparameters or build a deeper model, we likely would've seen a higher accuracy score closer to that of the CNN rather than the preliminary models.

While our results were extremely convincing, we also acknowledge that our dataset was very small compared to most datasets used to train effective machine learning models. One of the first and most important extensions to this project would be to test the CNN model on a larger set of test instances to get a better idea of how well it generalizes to new images. This process would also help us determine whether grid searching for the optimal model hyperparameters is worth the extra computing power necessary; if the model still performs well, there's no need to optimize it further. However, if it begins to perform poorly on a larger test set, it would be crucial to test different combinations of hyperparameters to ensure it remains effective even when generalizing to a larger set of potential input images.

VI. CONCLUSION

Our research successfully demonstrated the potential of advanced machine learning models in automating the classification of damage within water infrastructure systems. Beginning with basic models such as logistic regression, decision trees, and random forests, we observed limited success in accurately identifying damages due to the complex nature of the dataset. Our more advanced neural network models proved to be significantly better at handling the complexity of the task and data, and our custom AdaBoosted convolutional neural network reached a 100% accuracy score on the test data. Our results illustrate the effectiveness of neural networks in handling complex image classification tasks and fulfill our goal to engineer an accurate damage recognition model.

[1] K. Dala, Defect dataset, `https://universe.roboflow.com/krum-dala/defect-xwqhr` (2022), visited on 2024-03-19.