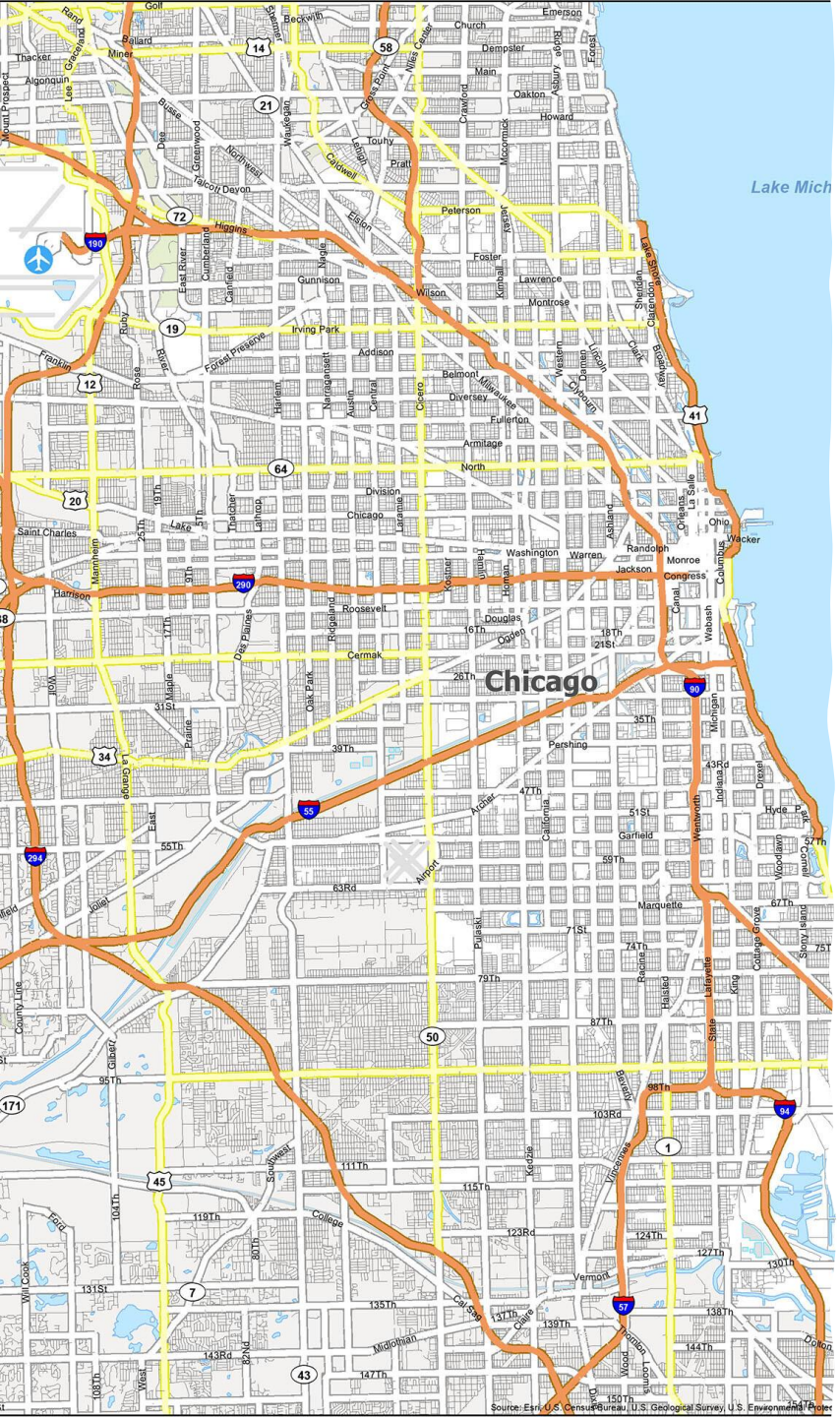
Image Classification for Prioritization of Pothole Repair

Marc Edwards

Machine Learning and Predictive Analytics

Spring 2023



Problem Statement

- Chicago has a over 8,000 miles of roads
- Lots of wear and tear – lots of potholes
  - Cause damage to vehicles, unsafe for pedestrians etc.
- Mapping and tracking of potholes manually reported to the City's 311 system
- No way to assess severity of the pothole, just a request to location using city's 311 system
- Image classification can be used for identifying and prioritizing pothole repairs
- Road maintenance crews can more efficiently allocate their resources and reduce the overall cost of and response time to repairs

Assumptions & Hypotheses

- Convolutional Neural Network (CNN) used for image classification
- We assume the images follow these assumptions before being able to use them in the image classification CNN:
  - Stationarity, Local Spatial Correlation, Translation Invariance, Feature Hierarchy
- We also formulate the following hypotheses about the data:
  - Discriminative features, Feature locality, Feature invariance, Generalization

Exploratory Data Analysis

- Data from Kaggle, Google Maps Street View, and collected Chicago street images
- 2798 images (.jpg files)
- Images belonged to 3 classes
  - notPothole (n = 370)
  - mildPothole (n = 449)
  - severePothole (n = 1979)
- 80/20 train-test split
  - 2238 images in train, 560 in test

Feature Engineering & Transformations

Several feature engineering steps applied to the images before use:

1. Rescale the Pixels in Images: images will be rescaled to a range between 0 and 1 (1./255)
2. Shear Transformations: shifting the position of pixels in a certain direction, creating a distorted effect
3. Zoom Ranges: magnifying or shrinking a specific portion of the image
4. Horizontal Flipping: random horizontal flipping of images, introduces additional variations in the train data
5. Resize Images (64x64 pixels): specifies the desired size to which the input images will be resized during the data loading process
6. Batch Size: the number of images in each batch generated

Proposed Approaches- CNNs vs. DNNs

CNNs have several advantages over fully connected Deep Neural Networks for this image classification problem.

- CNNs are much more parameter-efficient than fully connected DNNs. In CNNs, the same set of weights is used across different regions of the input image, which greatly reduces the number of parameters required. This is important for large-scale image classification problems, where the number of parameters can quickly become overwhelming.
- CNNs are designed to capture spatial hierarchies in the input data, which is crucial for image classification tasks. This allows the CNN to capture complex spatial relationships between features, such as edges and textures, which are important for accurate image classification.
- CNNs are translation invariant, which means that they can recognize objects regardless of their position in the image. This is achieved by using pooling layers, which downsample the feature maps and make them invariant to small translations in the input image.
- CNNs are also able to apply regularization techniques such as dropout and weight decay more effectively than fully connected DNNs.

Model Selection - Regularization

Sequential Model utilized for this CNN.

1. The first layer added to the model is a **Conv2D layer**, which performs convolution operations on the input image. It has 32 filters with a kernel size of 3x3. The activation function used is ReLU (Rectified Linear Unit), which introduces non-linearity.

2. After each Conv2D layer, a **MaxPooling2D layer** is added to reduce the spatial dimensions of the output feature maps, effectively downsampling the feature maps.

3. Another **Conv2D layer** with 64 filters and a kernel size of 3x3 is added, followed by another **MaxPooling2D layer**. These layers further extract higher-level features from the input data.

4. The **Flatten layer** is used to convert the 2D feature maps into a 1D feature vector, which can be fed into the subsequent fully connected layers.

5. Two fully connected (Dense) layers are added after the Flatten layer. The first **Dense layer** has 128 units/neurons with ReLU activation, allowing the model to learn more complex representations. The final **Dense layer** has 3 units, representing the number of classes in the classification task. It uses the softmax activation function, which produces a probability distribution over the classes.

6. The model is compiled using the **'adam' optimizer**, which is a popular choice for gradient-based optimization algorithms. The loss function used is ‘binary_crossentropy’. The metrics specified for evaluation during training are 'accuracy'.

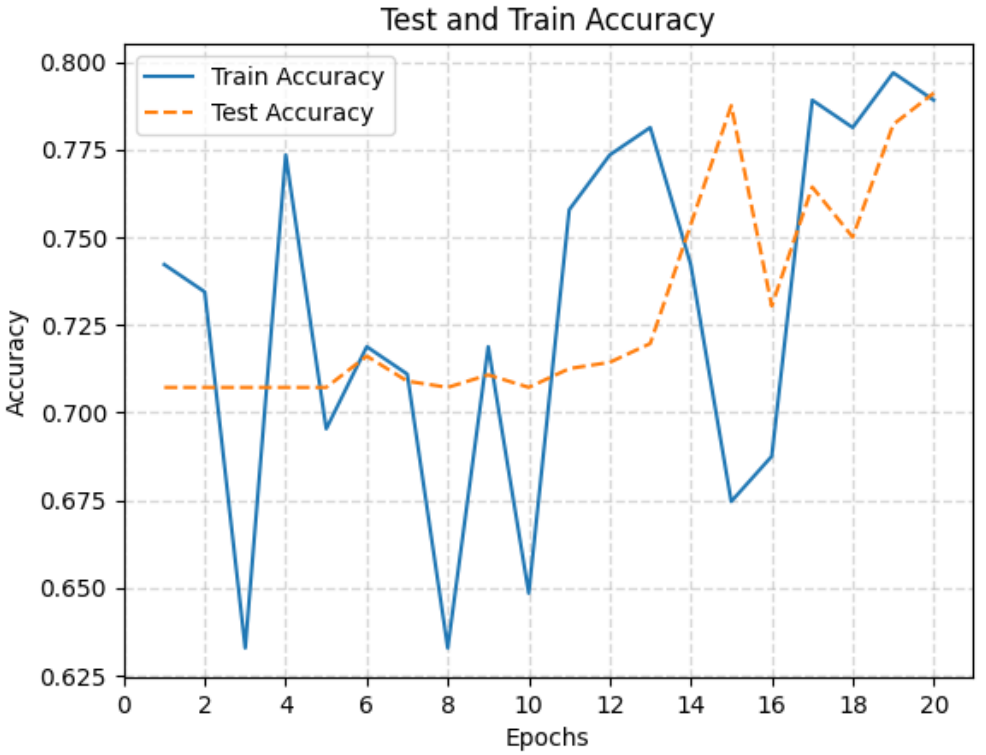
Model Selection

- After review of training errors and loss values, will fit 20 epochs and 4 steps per epoch for the CNN.
- Increasing the number of epochs increases the opportunities to learn from the data, and to fine-tune its parameters.
  - If the number of epochs is too high it can overfit the data, and not generalize well on the test.
  - Must find the right balance when increasing the number of epochs to not overfit.
- Increasing the steps per epoch increases the batches of data that will be processed in each epoch during the training process.
  - Increases the number of batches processed which can help it to learn more effectively.
  - Model will converge faster.
  - Results in higher accuracy and lower loss values for both the train and test sets.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)
 )

 conv2d_1 (Conv2D)           (None, 29, 29, 64)

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 64)
 2D)

 flatten (Flatten)           (None, 12544)

 dense (Dense)               (None, 128)

 dense_1 (Dense)             (None, 3)

=================================================================
Total params: 1,625,539
Trainable params: 1,625,539
Non-trainable params: 0
_________________________________________________________________
```

Results - Accuracy: 0.7643

- As epochs increase, so does the train and test accuracy, generally.
- There are not any signs of overfitting on the train data.
  - The accuracy score for the test data is slightly, but not significantly lower than the train accuracy across most of the epochs.
- The model is not overfitting on the train and is doing a great job of generalizing on the test data.



Key Takeaway: With an accuracy of 0.7643, a CNN model for pothole classification into notPothole, mildPothole, and severePothole classes is a viable solution when assessing the severity of potholes, which can aid street departments in prioritizing areas, efficiently allocate their resources, and reduce the overall cost of repairs.

Future Work

- More representative data – equal ratios of image classes
- Vision Transformer (VIT) versus Convolutional Neural Networks
- Live data collection from city vehicles
  - Attach cameras to city vehicles, utilize CV to capture road status data to quickly assess any holes to prioritize areas and efficiently allocate their resources and reduce the overall cost of repairs
- Pothole maintenance route planning

Thank you

References

- https://www.kaggle.com/datasets/rajdalsaniya/pothole-detection-dataset
- https://www.kaggle.com/datasets/atulyakumar98/pothole-detection-dataset
- https://www.kaggle.com/code/bsanandu88/pothole-detection-code-python