

Final Project (Week 6)

Project Description: Pothole Detection using Deep Learning

Ujwal Kumar

Vineet Upadhyay

Northeastern University

EAI6080: Adv. Analytical Utilization

Prof. Mimoza Dimodugno

05/10/2023

**Introduction**

Potholes are a major cause of accidents and damage to vehicles on roads. Early detection of potholes can help prevent accidents and save lives. This project aims to use deep learning techniques to detect potholes from images captured by a camera mounted on a moving vehicle.

**Technologies Used**

The following technologies were used in this project:

Python: The programming language used to write the code for this project.

TensorFlow: An open-source machine learning framework developed by Google that was used to implement the deep learning model for pothole detection.

Keras: An open-source neural network library written in Python that was used to build the deep learning model.

NumPy: A Python library used for numerical computing, which was used to manipulate the data for training and testing the model.

Matplotlib: A Python library used for data visualization, which was used to plot the accuracy and loss of the model during training.

Scikit-learn: A Python library used for machine learning tasks, which was used to train and evaluate machine learning models for pothole detection.

**Benefits**

Early detection of potholes can help prevent accidents and save lives.

The use of deep learning techniques can automate the process of pothole detection, making it more efficient and accurate than manual detection methods.

The trained model can be used for real-time pothole detection on moving vehicles, providing a cost-effective solution for pothole detection and road maintenance.

**Drawbacks and Challenges**

The availability and quality of training data can affect the accuracy of the model. In this project, the training data was limited to a specific dataset, which may not generalize well to other scenarios.

The performance of the model can be affected by factors such as lighting conditions, camera quality, and road surface conditions.

The deep learning model requires significant computational resources, which may not be available on low-powered devices.

**Working Example**

The project uses a convolutional neural network (CNN) to detect potholes in images. The CNN is trained on a dataset of images containing potholes and non-potholes. The model is trained to classify images as either potholes or non-potholes.

Once the model is trained, it can be used to classify new images as either potholes or non-potholes. In this project, the model is tested on a separate dataset of images and evaluated using various performance metrics such as accuracy, precision, recall, and F1-score.

In addition to the deep learning model, machine learning algorithms such as Random Forest Classifier, Support Vector Machines, and K-Nearest Neighbors are also used to classify the images and evaluate their performance. The results of these algorithms are compared with the results of the deep learning model to determine the most effective approach for pothole detection.

**Conclusion:**

Pothole detection using CNNs is a promising application of computer vision in road maintenance. It can improve the efficiency of manual inspection and potentially save lives by preventing accidents caused by potholes. However, the challenges associated with the quality of input images, availability of labeled data, and diversity of road conditions must be considered when developing and deploying such models.

Importing the essential libraries and modules, such as Keras, NumPy, Matplotlib, OpenCV, pandas, and scikit-learn, is the first step in the code. These libraries provide data processing, model development, visualization, and evaluation functionality.

The imagepaths list is then defined in the code to contain the paths of the images used for training and testing the model. It traverses the supplied directory and its subdirectories using the os.walk() function, collecting image paths and appending them to the imagepaths list.

The code then loads the photos and resizes them to a fixed size (IMG\_SIZE=128) before storing them in the X list. It assigns labels ('NEGATIVE' or 'POSITIVE') to each image based on its directory position and stores the labels in the y list at the same time.

To get insight into the dataset, the code uses Matplotlib's subplot feature to plot a random selection of photos along with their accompanying labels. This visualization aids in comprehending the dataset's distribution and properties.

The code performs the necessary preprocessing procedures before starting with model training. It encodes the labels with scikit-learn's LabelEncoder and changes them to one-hot encoded format with Keras' to\_categorical. This conversion is required for multi-class classification jobs involving 'NEGATIVE' and 'POSITIVE' classes.

The X and Y data are transformed to NumPy arrays, and the dataset is divided into training and testing sets using scikit-learn's train\_test\_split function. The testing set accounts for 25% of the whole data set.

The code then describes the architecture of the deep learning model by utilizing Keras' Sequential API. Multiple convolutional layers are followed by max-pooling layers, dropout regularization, and dense layers in the model. The final layer contains two units with softmax activation, one for each class ('NEGATIVE' and 'POSITIVE').

The Adam optimizer, categorical cross-entropy loss function, and accuracy metric are used to build the model. The model summary is printed to give a high-level overview of the model's design and number of trainable parameters.

Training is done by using the model's fit() method and handing it the training and testing data, the number of epochs, and the batch size. The training measurements are captured in the history object and can be utilized for further analysis and visualization.

The code assesses the model's performance on the testing set after training, estimating the loss and accuracy. The percentage of test accuracy is displayed.

After accuracy, the model predicts the labels for the testing set and converts the anticipated labels to their appropriate classes using np.argmax(). To visualize and analyze the model's training and validation performance, use the history.history.keys() function to access the keys (metrics) stored in the history object.

The below code generates two sets of charts to evaluate the trained model's performance. The first set depicts improvements in accuracy and loss during the training phase, while the second set concentrates on the validation (testing) set's accuracy and loss metrics.

The function starts by converting the y\_test data into binary format, which represents the positive and negative classes. The confusion matrix is then displayed in a pandas DataFrame, which shows the counts of true positive, true negative, false positive, and false negative predictions.

The code then uses the reshape() function to convert the training and testing data into flat arrays. This is done to prepare the data for machine learning classifier training and prediction.

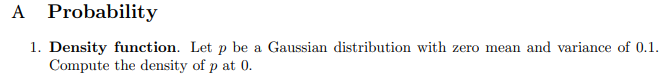
The code then evaluates the Random Forest Classifier's performance. It uses flattened training data to train the classifier and flattened testing data to make predictions.

The code also assesses the Support Vector Machines (SVM) classifier by training it on flattened training data and predicting it on flattened testing data.

Finally, the algorithm performs an evaluation of the K-Nearest Neighbors (KNN) classifier. It uses flattened training data to train the classifier and flattened testing data to predict.

The algorithm calculates the accuracy score for each classifier and creates a classification report that includes metrics such as precision, recall, and F1-score for each class.

The generated results and reports for each classifier can be compared to see how well they detect potholes. This information is critical for selecting the best classifier for the task at hand.



Solution: The following formula represents the probability density function of a Gaussian distribution with mean and variance σ^2:

f(x) = (1/√(2πσ^2)) \* e^(-(x-μ)^2 / 2σ^2)

We have a Gaussian distribution in this situation, with a variance of 0.1 and a mean of 0.

μ = 0

σ^2 = 0.1

Substituting these values in the equation above, we get:

f(x) = (1/√(2π0.1)) \* e^(-(x-0)^2 / 20.1)

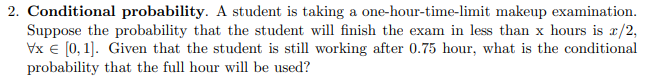
Simplifying further, we get:

f(x) = (1/√(0.2π)) \* e^(-(x^2) / 0.2)

Now, we need to compute the density of p at 0, i.e., we need to find f(0):

f(0) = (1/√(0.2π)) \* e^(-(0^2) / 0.2) = (1/√(0.2π)) \* e^0 = (1/√(0.2π)) = 1.987

Therefore, the density of p at 0 is 1.987

Solution: We can apply Bayes' theorem to find a solution to this issue. Let A represent the scenario in which the student completes the exam in the allotted time, and B represents the scenario in which the student is still working after 0.75 hours. The next step is to locate P(A|B).

The Bayes theorem provides us with:

P(A) = P(B) / P(B) = P(A|B)

Since the student must still be working after 0.75 hours if they complete the exam in the allotted time, we can conclude that P(B|A) = 1. Since the likelihood that a student would finish the exam in exactly one hour is half, we also know that P(A) = 1/2. This is because the chance that a student will finish the exam in fewer than x hours is x/2 for x ∈ [0, 1].To find P(B), we can use the law of total probability:

P(B) = P(B|A)P(A) + P(B|A')P(A')

where A' stands for the possibility that the student completes the test in less time than an hour. We can conclude that P(B|A') = 0.25 because the likelihood that the student will finish in less than one hour, given that they do so in less than one hour, is 1/2, and the probability that they will finish in between 0.75 and 1 hour is 0.25 of that.

Therefore, we have:

P(B) = 1*1/2 + 0.25*1/2 = 0.625

Plugging all of these values into the formula for P(A|B), we get:

P(A|B) = 1\*1/2 / 0.625 = 0.8

In light of the fact that the student is still working after 0.75 hours, the conditional chance that they will use the entire hour is 0.8.

Graphical user interface, text, application

Description automatically generatedSolution: We want to find P(sick = yes | Weather = rainy), which is the probability of getting sick given that it is raining. We can use Bayes' rule to calculate this:

P(sick = yes | Weather = rainy) = P(Weather = rainy | sick = yes) \* P(sick = yes) / P(Weather = rainy)

From the table, we know that P(Weather = rainy) = 0.08 + 0.02 = 0.1, and P(sick = yes) = 0.144 + 0.02 + 0.016 + 0.02 = 0.2.

To find P(Weather = rainy | sick = yes), we need to use the formula:

P(Weather = rainy | sick = yes) = P(sick = yes | Weather = rainy) \* P(Weather = rainy) / P(sick = yes)

We can rearrange this to get:

P(sick = yes | Weather = rainy) = P(Weather = rainy | sick = yes) \* P(sick = yes) / P(Weather = rainy)

P(sick = yes | Weather = rainy) = (0.02 / 0.2) \* 0.2 / 0.1

P(sick = yes | Weather = rainy) = 0.02 / 0.1

P(sick = yes | Weather = rainy) = 0.2

Therefore, the conditional probability of getting sick given that it is raining is 0.2

Graphical user interface, Word

Description automatically generated with medium confidenceSolution: We start by using the chain rule:

Let u = 1 + e^-z, then f(z) = 1/u.

Now, we can use the power rule and the chain rule:

df/dz = d/dz (u^(-1)) = -u^(-2) \* du/dz

To find du/dz, we use the chain rule again:

Let v = -z, then u = 1 + e^v.

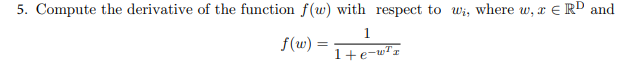
Then,

du/dz = du/dv \* dv/dz = e^v \* (-1) = -e^(-z)

Substituting this into the previous equation, we get:

df/dz = -1/(1+e^(-z))^2 \* (-e^(-z)) = e^(-z) / (1+e^(-z))^2

Therefore, the derivative of f(z) with respect to z is e^(-z) / (1+e^(-z))^2.



Solution: Using the chain rule of differentiation, we have:

∂f(w)/∂wi = ∂f(w)/∂(w^Tx) \* ∂(w^Tx)/∂wi

where ∂f(w)/∂(w^Tx) is the derivative of the function f(w) with respect to w^Tx, and ∂(w^Tx)/∂wi is the derivative of the inner product w^Tx with respect to the ith element of w.

From the given function, we have:

f(w) = 1/(1 + e^(-w^Tx))

Taking the derivative of f(w) with respect to w^Tx:

∂f(w)/∂(w^Tx) = e^(-w^Tx) / (1 + e^(-w^Tx))^2

Now, taking the derivative of the inner product w^Tx with respect to wi:

∂(w^Tx)/∂wi = xi

Therefore, we have:

∂f(w)/∂wi = ∂f(w)/∂(w^Tx) \* ∂(w^Tx)/∂wi = e^(-w^Tx) / (1 + e^(-w^Tx))^2 \* xi

Thus, the derivative of f(w) with respect to wi is e^(-w^Tx) / (1 + e^(-w^Tx))^2 times the ith element of x.

Word

Description automatically generated with medium confidenceSolution: The derivative of the loss function J(w) with respect to w can be computed as follows:

For each component wi of w, we have:

dJ(w)/dwi = Σ(i=1 to m) d/dwi (1/2 |w^Tx^(i) - y^(i)|)

We can simplify the absolute value expression by introducing a new variable ei:

ei = w^Tx^(i) - y^(i)

Then we have:

|ei| = sqrt(ei^2)

Now we can differentiate the expression inside the summation:

d/dwi (1/2 |ei|) = 1/2 d/dwi sqrt(ei^2) = 1/2 (ei/|ei|) d/dwi ei

The derivative d/dwi ei is simply the i-th component of x^(i), which we denote as xi,i. Therefore:

d/dwi (1/2 |w^Tx^(i) - y^(i)|) = 1/2 (ei/|ei|) xi,i

Substituting back ei = w^Tx^(i) - y^(i), we get:

d/dwi (1/2 |w^Tx^(i) - y^(i)|) = 1/2 ((w^Tx^(i) - y^(i))/|w^Tx^(i) - y^(i)|) xi,i

Finally, we can put it all together to obtain the derivative of the loss function with respect to w:

dJ(w)/dwi = Σ(i=1 to m) d/dwi (1/2 |w^Tx^(i) - y^(i)|) = Σ(i=1 to m) 1/2 ((w^Tx^(i) - y^(i))/|w^Tx^(i) - y^(i)|) xi,i

Simplifying further, we can write:

dJ(w)/dwi = Σ(i=1 to m) ((w^Tx^(i) - y^(i))/|w^Tx^(i) - y^(i)|) xi,i

Note that the expression (w^Tx^(i) - y^(i))/|w^Tx^(i) - y^(i)| is equal to the sign of the residual ei. If ei is positive, the sign is 1, and if ei is negative, the sign is -1. Therefore, the derivative is proportional to the direction of the residual, and the magnitude is equal to the absolute value of the residual times the i-th component of x^(i).



Graphical user interface, text, email

Description automatically generated

Solution:

a. **ReLU**:

Equation: f(x) = max(0, x)

Derivative: f'(x) = 1 if x > 0, f'(x) = 0 if x <= 0

Chart, line chart

Description automatically generated

(Figure 1: ReLU)

b. **Tanh**:

Equation: f(x) = tanh(x) = (e^x - e^-x) / (e^x + e^-x)

Derivative: f'(x) = 1 - tanh^2(x)

Chart, line chart

Description automatically generated

(Figure 2: Tanh)

c. **Softmax**:

Equation: f(x\_i) = e^(x\_i) / Σ\_j e^(x\_j)

Derivative: f'(x\_i) = f(x\_i)(1 - f(x\_i)) for i = j, and f'(x\_i) = -f(x\_i)f(x\_j) for i ≠ j

A picture containing chart

Description automatically generated

(Figure 3: Softmax)

d. **Sigmoid**:

Equation: f(x) = 1 / (1 + e^-x)

Derivative: f'(x) = f(x)(1 - f(x))

Chart, line chart

Description automatically generated

(Figure 4: Sigmoid)

e. **Leaky ReLU**:

Equation: f(x) = max(ax, x)

Derivative: f'(x) = a if x < 0, f'(x) = 1 if x >= 0

Chart, line chart

Description automatically generated

(Figure 5: Leaky ReLU)

f. **Sinc**:

Equation: f(x) = sin(x) / x

Derivative: f'(x) = (x cos(x) - sin(x)) / x^2 if x != 0, f'(x) = 0 if x = 0

Chart, line chart

Description automatically generated

(Figure 6: Sinc)

**D - References**

1. H., Asad, M. H., Khaliq, S., Yousaf, M. H., Ullah, M. O., & Ahmad, A. (2022, April 20). *Pothole Detection Using Deep Learning: A Real-Time and AI-on-the-Edge Perspective*. Pothole Detection Using Deep Learning: A Real-Time and AI-on-the-Edge Perspective. https://doi.org/10.1155/2022/9221211
2. *Convolutional neural networks based potholes detection using thermal imaging*. (2019, February 5). Convolutional Neural Networks Based Potholes Detection Using Thermal Imaging - ScienceDirect. https://doi.org/10.1016/j.jksuci.2019.02.004
3. H., & Hoang, N. D. (2018, June 6). *An Artificial Intelligence Method for Asphalt Pavement Pothole Detection Using Least Squares Support Vector Machine and Neural Network with Steerable Filter-Based Feature Extraction*. An Artificial Intelligence Method for Asphalt Pavement Pothole Detection Using Least Squares Support Vector Machine and Neural Network With Steerable Filter-Based Feature Extraction. https://doi.org/10.1155/2018/7419058
4. Sanad, M. (2020, February 18). *Image Classification Using CNN (Convolutional Neural Networks)*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/02/learn-image-classification-cnn-convolutional-neural-networks-3-datasets/
5. *Potholes caused over 5,000 deaths, shows 2018-2020 road accidents data*. (2022, August 24). The Federal. https://thefederal.com/news/potholes-caused-over-5000-deaths-shows-2018-2020-road-accidents-data/
6. *Self-Driving Technology Represented in Art*. (n.d.). IEEE Potentials Magazine, January/February 2020Self-Driving Technology Represented in Art. https:////read.nxtbook.com/ieee/potentials/january\_february\_2020/self\_driving\_technology\_repre.html
7. Ahmed, K. R. (2021, December 16). *Smart Pothole Detection Using Deep Learning Based on Dilated Convolution*. PubMed Central (PMC). https://doi.org/10.3390/s21248406
8. Mishra, M. (2020, September 2). *Convolutional Neural Networks, Explained*. Medium. https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939
9. Sanad, M. (2020, February 18). *Image Classification Using CNN (Convolutional Neural Networks)*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/02/learn-image-classification-cnn-convolutional-neural-networks-3-datasets/
10. *Conditional probability - Wikipedia*. (2020, September 11). Conditional Probability - Wikipedia. https://en.wikipedia.org/wiki/Conditional\_probability