Comparison of Different Convolutional Neural Network Architecture for Pothole Detection

Jeyanth K. Ramasamy
Post-graduate Student
Department of Computer Science
Dalhousie University
Halifax, NS, Canada B3H 4R2
jy253364@dal.ca

Janvi Patel
Post-graduate Student
Department of Computer Science
Dalhousie University
Halifax, NS, Canada B3H 4R2
jn410076@dal.ca

Robinder J. Dhillon
Post-graduate Student
Department of Computer Science
Dalhousie University
Halifax, NS, Canada B3H 4R2
rb802397@dal.ca

Vishal U. Sancheti
Post-graduate Student
Department of Computer Science
Dalhousie University
Halifax, NS, Canada B3H 4R2
vs488310@dal.ca

Abstract

Road maintenance has always been a challenging task. Year after year, the accident rates are increasing due to the up-surging potholes count. As the road maintenance process is done manually in most places, it consumes enormous time, requires human labor, and is subjected to human errors. Thus, there is a growing need for a cost-effective automated identification of potholes. Many approaches proved good results in applying deep learning [1] for different object detection. Convolutional Neural Networks (CNNs) can learn the art of extracting relevant features from an Image. The data set consisting of 510 training images, and 171 testing images of normal road and pothole is trained on VGG, AlexNet, ResNet, LeNet, and the results are compared. The model is then tested on different pothole images, and it detects with reasonable accuracy.

1 Introduction

The term 'road' for land transportation infrastructure covers all parts of the road, including complementary buildings and traffic equipment. In transportation to support safety, security, and comfort, a decent and good quality road is needed. One indicator of road feasibility is whether or not there are potholes on the road. Traditionally, one method used to obtain data on the number of potholes on the road is based on reports from the surrounding community. To improve the data gathering method, we suggest creating a system that detects potholes on the road using a Convolutional Neural Network.

The pothole is detected by acquiring the image of the highway, which is then convoluted with the convolution matrix so that the dimensions of the image matrix become smaller without losing the image characteristics. Then the convolution results are trained with data sets using the neural network to issue a decision whether the pothole is detected or not. Detection of potholes is much more difficult when compared to other objects such as pedestrians, vehicles, traffic signs, etc., because the former has a wide range of geometric. When comparing the recognition algorithms, Convolutional Neural Networks (CNN) has proven to be one of the best [2].

The purpose of this study is to identify the best convolutional neural network model suited to detect potholes from an image. In further research, the system can apply this processing technique to real-time video to overcome problems in monitoring and evaluating roads.

2 Related Work

There are so many architectures and algorithms that can be used for potholes detection. Evaluating a pothole in most cases requires 3D equipment, which is very expensive and not feasible on a large scale [3]. Instead, detecting a defect in the road using an image and then further processing it into defective and not-defective regions and analyzing the patterns in both the regions can help differentiate a pothole from the non-defective region, which is essentially the road.

Another method of analyzing these problems is using three features of an image: the texture, shape, and dimensions of the defective area in the frame [4], [5]. But the method does not have machine learning at the core of it. Therefore, one of the most basic ways to carry out potholes detection is using Convolutional Neural Networks (CNNs).

The CNN-based method consists of two stages. The first stage is the classification of images using feed-forward, and the second stage is the learning stage with the backpropagation method. Before classification, preprocessing is done and then trained using the feed-forward and backpropagation methods. Finally, the classification stage uses the feed-forward method with updated weights and biases.

Earlier, some of the Object Detection fields started with Region-Based Convolutional Neural Networks (R-CNN) and moved to faster and more advanced algorithms like YOLO (You Only Look Once) and SSD (Single-Shot Detection). The R-CNNs use selective search algorithms to extract 2000 bounding boxes from the input image in the first step itself to stick with processing only the most essential features in an input based on color, pattern, shape, and size [6].

The R-CNNs use a three-stage mechanism: feature extraction via Selective Search Algorithm, SVM classification, and Regression modeling for tight bounding boxes. The Fast R-CNN model uses a single-stage mechanism where it directly passes the input to a CNN, and the output from this CNN is the Regions of Interest (RoI) [7]. Then an RoI pooling layer is applied to the CNN's output to warp the images to the size the Network is accustomed to. These RoIs are given to a fully connected Neural Network (NN) that segregates them and returns bounding boxes on the RoIs using linear regression and softmax networks working in a parallel manner and provides drastic speed gains as well as savings in terms of the size of the model.

Although Fast R-CNN uses a one-stage mechanism, it relies on the selective search method initially, which consumes time. Hence, trying out CNN-based models is a much more efficient way we have implemented in this work and compared performance of multiple CNN-based models such as VGG, AlexNet, ResNet, and LeNet.

3 Data and Methodology

3.1 Data Pre-Processing

For our project, We downloaded images from a data set on Kaggle. Initially, we checked if all images were colored or grayscale. Since grayscale images' proposition was less than 3%, we removed them, and now the dataset has only colored images. We resized and compressed all the images so that the dimensions and pixel intensity were the same for all images. These pre-processed images are converted to numerical data. Colored images have three channels denoting R, G, B colors. Since the range of each pixel in the channel is 0-255, normalization is carried out so that the dataset can use a standard scale. The dataset is then divided into training and test datasets in the ratio of 7:3, 70% being the training data and 30% being test data.

3.2 Data Operations

All the images are resized accordingly to model architecture. The image data is stored in a NumPy array. Correspondingly, the labels ("NORMAL" and "POTHOLE") of the images are stored in another array. LabelEncoder is used to encode target labels with values between 0 and n_classes-1. The label array is transformed into categorical values. The data array is normalized for better approximation.

We have used image augmentation techniques such as shifting, rotating, and flipping. Each copy is unique in certain respects. When the model is trained on new, slightly altered images, it becomes more resilient. Further, the learning rate for the models is reduced during run-time using the ReduceLROnPlateau callback. Loss functions, activation functions, and optimizers are decided to depend on the model's model architecture and outcome.

3.3 Pothole Detection using LeNet

The LeNet network has five layers with learnable parameters and hence named Lenet-5. LeNet has seven layers which comprise the convolutional layer, pooling layer, and full connection layer. In a CNN, convolutional layers are typically arranged so that they gradually decrease the spatial resolution of the representations while increasing the number of channels. It has an input layer with a feature map size of 32X32X1. Then it is followed by an alternate convolution layer and average pooling layer. To handle the vanishing gradient problem, we have used the softmax activation function for the dense layer. The loss function was altered from binary_crossentropy to categorical_crossentropy to compute the cross-entropy loss between the labels and predictions. Finally, fine-tuning of the model was carried out to achieve better accuracy.

3.4 Pothole Detection using AlexNet

AlexNet has a similar architecture as that of LeNet, but it has more filters per layer and has stacked convolutional layers. The architecture consists of five convolution layers and three fully connected layers. Input to this model is the images of size 227X227X3. AlexNet uses ReLU instead of tanh, which accelerates the model's speed to six times with the same accuracy. It uses a dropout layer with an input unit of 0.5. It uses multiple GPUs; half of the model's neurons in one GPU and the other half on another GPU which helps in cutting training time, and a bigger model can be trained. Similar to LeNet, it uses categorical_crossentropy loss function and softmax activation function.

3.5 Pothole Detection using VGG

VGG Neural Networks. While previous derivatives of AlexNet focused on smaller window sizes and strides in the first convolutional layer, VGG addresses another critical aspect of CNNs: depth. VGG uses ReLu as an activation function which increases the efficiency. VGG takes in a 224x224 pixel RGB image. VGG has three fully connected layers: the first two have 4096 channels each, and the third has 1000 channels, 1 for each class. All of VGG's hidden layers use ReLU (a tremendous innovation from AlexNet that cuts training time). The convolution stride is fixed to 2 pixels so that the spatial resolution is preserved after convolution. Max-pooling is performed over a 2×2 pixel window, with stride 2. VGG does not generally use Local Response Normalization (LRN), as LRN increases memory consumption and training time with no particular increase in inaccuracy.

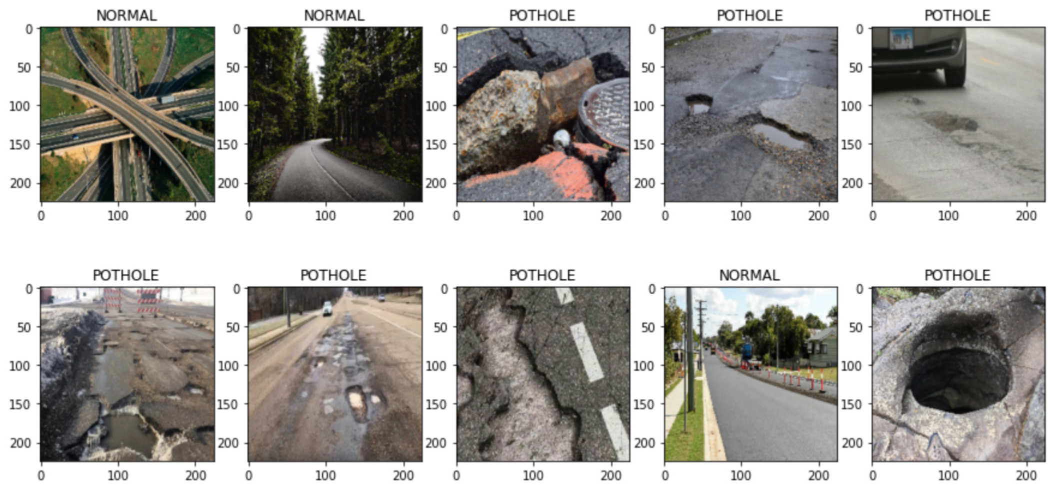
3.6 Pothole Detection using ResNet

Residual Neural Network is an artificial neural network whose core idea is an "identity shortcut connection" that skips one or more layers. The skipped layers are then restored as it learns the feature space. Learning speed increases as skipping reduces the impact of the vanishing gradient. Since ResNet has a deep network, the gradient from which loss function is calculated reduces to zero after several applications of the chain rule. ResNet consists of one convolution and pooling step followed by 4 layers of similar behavior. Each of the layers follows the same pattern. They perform 3x3 convolution with a fixed feature map dimension (F) [64, 128, 256, 512] respectively, bypassing the input every 2 convolutions. We define the model using tensorflow's prebuilt ResNet50 and replacing the final fully-connected layer with a dense layer of 4096 nodes with ReLU activation. It is then followed by dense and dropout layers with required parameters. Like other models, it uses categorical_crossentropy loss function and Adam optimizer.

4 Experiments

Various experiments were conducted on the selected dataset. For the purpose of the experiment, images were resized either to 227x227 or 32x32, depending on the model. Also, they were labeled with 0 and 1 depending on the directory they were placed in, i.e., either Normal or Pothole. These images and labels finally generated a training dataset of 510 images and a test dataset of 170 images and also made sure they both did not have the same image.

Figure 1: Random Samples from Pre-Processed and Labelled Dataset



Execution of all the selected CNNs carried out on CPU-based engine using google collab platform either on hosted run-time or local run-time using jupyter notebook.

4.1 Model Tuning

Initially all the selected CNNs were tuned individually to improve performance. The various actions performed are as follow:

4.1.1 LeNet

Table 1: Model Tuning on LeNet

Actions Performed		
	Description	Accuracy (%)
1	Initialized with tanh activation function and SGD optimizer with lr=0.1	∼49.5
2	Introduced ReduceLRonPlateau and changed lr to 1e-6	∼50.5
3	Changed lr to 5e-4	∼48.4
4	Changed activation function from tanh to ReLU and lr to 0.2	∼68.6
5	Changed optimizer from SGD to Adam and lr to 0.01	∼82.4
6	Changed lr to 5e-4	∼84.7

4.1.2 AlexNet

Table 2: Model Tuning on AlexNet

	Actions Performed	
	Description	Accuracy (%)
1	Initialized with ReLU activation function and SGD optimizer with lr=0.2	∼49.41
2	Introduced ReduceLRonPlateau and changed lr to 1e-6	∼50.1
3	Changed lr to 5e-4	∼52.54
4	Changed optimizer from SGD to Adam	∼82.94

4.1.3 VGG

Table 3: Model Tuning on VGG

	Actions Performed	
	Description	Accuracy (%)
1	Initiated with neural nodes input same as VGG16 architecture	52
2	Changed activation function “tanh” from “relu” and architecture*	50
3	Changed drop out to 0.4	81
4	Changed drop out to 0.3 and epochs to 29 from 30	88
5	Changed epochs to 100 epochs	82.68
6	Changed activation function to “sigmoid” from “softmax” in dense layer	53
7	Changed optimizer from Adam to SGD	48
8	Added kernel regularizer = ‘l2’ as fine tune parameter	90
9	Changed activation function by “elu”	90

* VGG16 and set neural inputs as 128 for convolution layers instead of 256, 512 inputs and dense layers’ input neuron has also been changed by 128

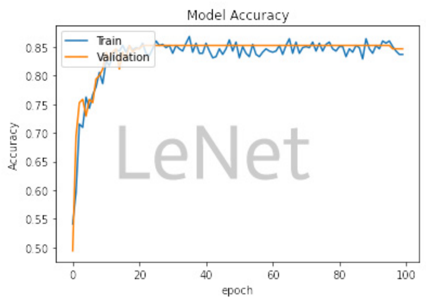
4.1.4 ResNet

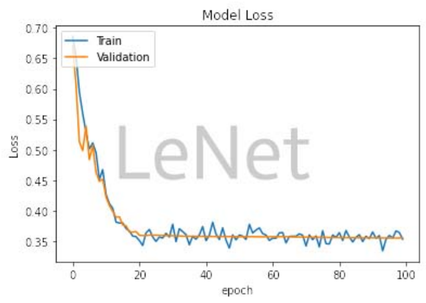
Table 4: Model Tuning on ResNet

	Actions Performed	
	Description	Accuracy (%)
1	Initialized with ReLU activation function and SGD optimizer with lr=0.2	∼52.12
2	Introduced ReduceLRonPlateau and changed lr to 1e-6	∼54
3	Changed lr to 5e-4	∼54.3
4	Changed optimizer from SGD to Adam	∼96

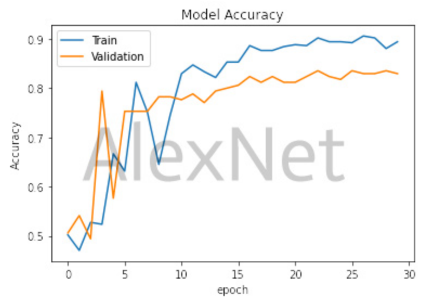
4.2 Model Results

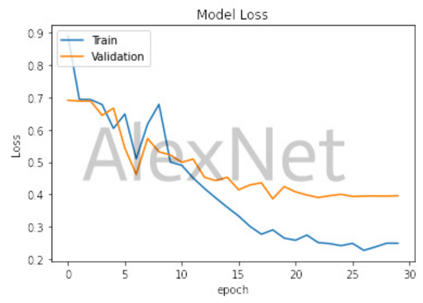
4.2.1 LeNet



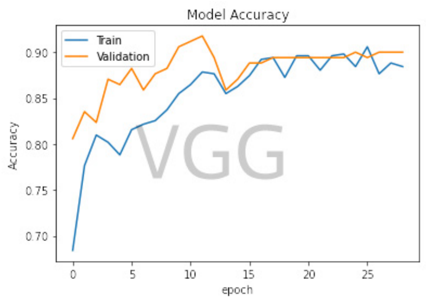


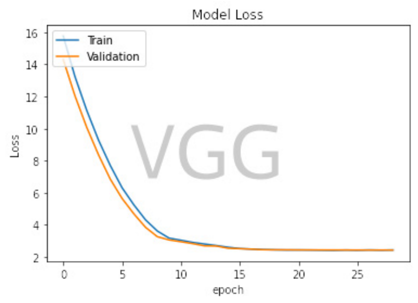
4.2.2 AlexNet



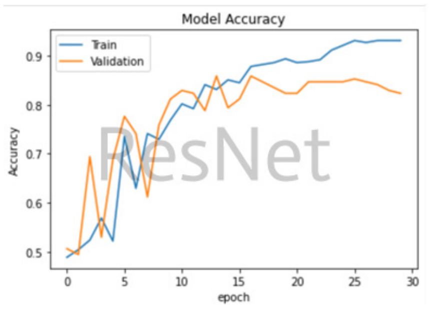


4.2.3 VGG





4.2.4 ResNet



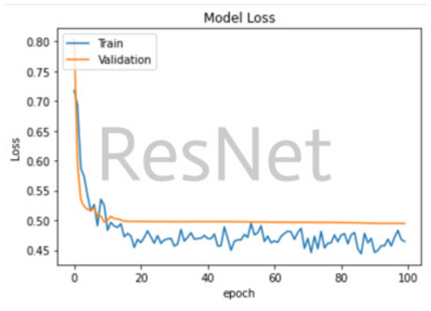
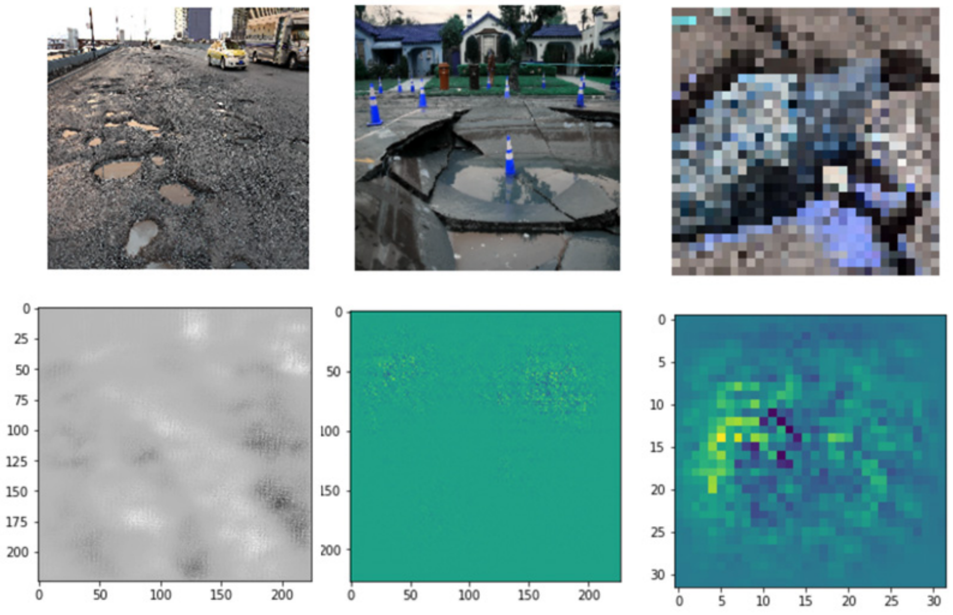


Table 5: Model Results

Model	Accuracy (%)	Confusion Matrix
LeNet	84.7(%)	[[68 16][10 76]]
AlexNet	82.94(%)	[[66 18][11 75]]
VGG	90(%)	[[72 9][3 83]]
ResNet	96(%)	[[79 5][1 85]]

Figure 2: Sample Visualization for Different Models



5 Learnings

5.1 Activation function and Loss function

In a neural network, an activation function specifies how the input's weighted number is converted into an output from a node or nodes in a layer. The activation function chosen has a significant impact on the neural network's capability and efficiency.

In our experiment, the choice of activation function included tanh, ReLU, and ELU for Conv2D layers and Sigmoid and Softmax for the Dense layer. It is observed that ReLU performed well for all of the models. For VGG, ELU was used, which showed higher performance. ELU is very similar to RELU except for negative inputs. They are both in identity function form for non-negative inputs. Similarly, Softmax performed well compared to Sigmoid.

5.2 Optimizer

Optimizers are used to tinker the weights to shape the model into the most reliable one, while the Loss acts as a road map for the optimizer directing it in the best way possible.

In our experiment, the models initially use SGD but eventually shifted to Adam. It utilizes the combined advantages of two other variants of SGD: the Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp).

5.3 ReduceLROnplateau

ReduceLROnPlateau reduces the learning rate if there is no improvement in a metric. Often training stagnates at some stage, and reducing the learning rate by a factor would benefit the model. It is observed accuracy of the models was improved to a greater extent using this callback.

5.4 Model

The experiment results are shown in Table 5. It suggests every CNN with proper tuning performs well with high accuracy. Thus, the quality of the algorithm could not be estimated with the only accuracy. To assess further, the introduction of Precision, Recall, F1 Score, and Dice score Coefficient is proper.

Here, Precision (P), Recall (R), F1 Score (F1), and Dice score Coefficient (DSC) can be given using following equations

$$P = TP/(TP + FP)$$

$$R = TP/(TP + FN)$$

$$F1 = 2(P * R)/(P + R))$$

$$DSC = 2TP/((TP + FP) + (TP + FN))$$

After calculating the newly introduced metrics we have following Table 6.

Table 6: Precision, Recall, F1, and DSC of Models

Model	Precision	Recall	F1	DSC
LeNet	0.8717	0.4722	0.6125	0.7453
AlexNet	0.8571	0.4680	0.6054	0.8170
VGG	0.96	0.4645	6260	0.9230
ResNet	0.9875	0.4817	0.6475	0.9634

Here, It can be observed that VGG and ResNet outperforms LeNet and AlexNet. Also, ResNet has better F1 and DSC compared to VGG.

6 Discussion and Conclusion

We implemented several experiments to identify the best CNN algorithm for pothole detection. We believe pothole detection will reduce accidents and improve human beings' safety for travel on the road.

Based on the experimental observations, we can also conclude that with some proper tuning, VGG and ResNet are best suited CNN models for pothole detection; moreover, ResNet performed the best among all CNN models with our dataset.

Apart from pothole detection on an image, this project can be enhanced to use live video stream with self-driving cars to identify the potholes on the road and alert the driver to reduce the speed in advance. It is even possible to invent a new technology that automatically reduces the speed of the vehicle based on the existence of potholes on the road.

Acknowledgement

We are grateful to Dalhousie University for providing us the support required for the experiment. We want to thank professor Sageev Oore for enhancing our knowledge of Machine Learning. And to mention Chandramouli Shama Sastry, who has contributed as a mentor in our project.

References

[1] Geronimo, David, et al. "Survey of pedestrian detection for advanced driver assistance systems." IEEE transactions on pattern analysis and machine intelligence 32.7 (2009): 1239-1258

[2] Du, Juan. "Understanding of Object Detection Based on CNN Family and YOLO." Journal of Physics: Conference Series. Vol. 1004. No. 1. IOP Publishing, 2018.

[3] Koch, Christian, and Ioannis Brilakis. "Pothole detection in asphalt pavement images." Advanced Engineering Informatics 25.3 (2011): 507- 515.

[4] Huidrom, Lokeshwor, Lalit Kumar Das, and S. K. Sud. "Method for automated assessment of potholes, cracks and patches from road surface video clips." Procedia-Social and Behavioral Sciences 104.2013 (2013):312-321.

[5] Karthika, R., and Latha Parameswaran. "An automated vision-based algorithm for out of context detection in images." International Journal of Signal and Imaging Systems Engineering 11.1 (2018): 1-8.

[6] Girshick, Ross, et al. "Region-based convolutional networks for accurate object detection and segmentation." IEEE transactions on pattern analysis and machine intelligence 38.1 (2015): 142-158.

[7] Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE international conference on computer vision. 2015.

[8] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[9] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[10] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.

[11] L.K. Suong, and K. Jangwoo, “Detection of potholes using a deep convolutional neural network,” J. Universal Computer Sci., vol. 24, pp. 1244-1257.

[12] R. Joseph, D. Santosh, G. Ross, and F. Ali, “You Only Look Once: Unified, Real-Time Object Detection,” pp. 779-788, 2016, 10.1109/CVPR.2016.91.