# Automated Bolt Fitting System with Computer Vision-Guided Robotic Arm

**M. Nandhini[1,*], Arya Patel[2], Meherzad Sukheswala[3] and Aryan Iyer[4]**
Department of Mechatronics Engineering, SRM Institute of Science and Technology, KTR

Email: nandhinm@srmist.edu.in

**Abstract:** This system aims to streamline and enhance the bolt fitting process in industrial settings by employing advanced computer vision techniques to accurately locate bolt positions and guide the robotic arm for precise fitting. By integrating machine vision algorithms with robotic manipulation, the system ensures efficient and error-free bolt fitting, reducing manual labor requirements and improving overall productivity. Key components of the proposed system include image processing algorithms for bolt detection and localization, robotic arm control for automated manipulation, and a user-friendly interface for seamless operation. The implementation of this system promises to revolutionize bolt fitting tasks in various industries, offering enhanced speed, accuracy, and reliability in assembly processes.

**Keywords:** Computer Vision, Robotic Arm, Automated Bolt

## 1. Introduction

In the realm of industrial manufacturing, efficiency and precision are paramount. However, traditional methods of bolt fitting often fall short of these ideals, relying heavily on manual labor that is prone to error and inefficiency. This inefficiency not only impacts production schedules but also introduces the potential for defects and safety hazards. Recognizing these challenges, the concept of automation has gained significant traction in recent years. Automation promises to revolutionize industrial processes by leveraging technologies such as robotics and computer vision to streamline operations and enhance productivity. One area ripe for automation is the fitting of bolts, a ubiquitous task in assembly processes across various industries. The manual fitting of bolts involves several steps, including locating bolt positions, aligning bolts with corresponding holes, and tightening them to specified torque levels. Each of these steps is subject to human error and can be time-consuming, particularly in large-scale manufacturing environments. Moreover, repetitive manual tasks increase the risk of ergonomic injuries for workers. To address these challenges, the "Automated Bolt Fitting System with Computer Vision-Guided Robotic Arm" project aims to develop an innovative solution that combines the power of computer vision and robotics. By integrating advanced imaging techniques with precise robotic manipulation, the project seeks to automate the bolt fitting process from start to finish. The use of computer vision technology enables the system to accurately detect bolt positions within the assembly area in real-time. This eliminates the need for manual inspection and measurement, saving valuable time and reducing the likelihood of errors. The robotic arm serves as the mechanical interface of the system, responsible for manipulating bolts with precision and efficiency. Equipped with control algorithms tailored to the task, the robotic arm can handle bolts, ensuring consistent and reliable tightening. Beyond improving efficiency and precision, the automated bolt fitting system prioritizes worker safety. By minimizing the need for manual intervention in physically demanding tasks, the system reduces the risk of ergonomic injuries and creates a safer work environment for operators. Furthermore, the "Automated Bolt Fitting System with Computer Vision-Guided Robotic Arm" project represents a significant step forward in the automation of industrial assembly processes. By leveraging cutting-edge technologies, the project aims to enhance productivity, accuracy, and safety, ultimately driving greater efficiency and competitiveness in manufacturing operations.

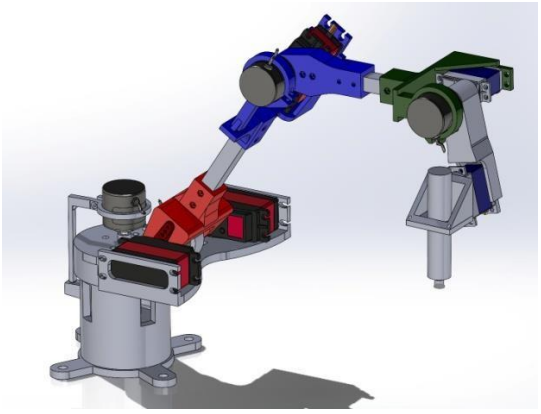


Figure 1. Robotic Arm CAD

## 2. Methodology

Using deep learning, the first step of creating a model with the help of neural networks involves the collection of datasets. The dataset collected will entail several images corresponding to different classes of existing road conditions. The classes that have been primarily identified and will be under study includes:

- Plain Roads
- Wet/Slurry Roads
- Snowy/Icy Roads

The architecture of the model that is being created comes in fruition with the help of transfer learning and fine-tuning. This involves the use of an already pre-trained model instead of creating one from scratch. With the addition and the help of fine-tuning, certain layers as well as parameters that already exist in the pre-trained network remain constant with the help of new ones. The combination of both techniques helps in creating a model that allows the objectives of this paper to be carried out. Training the resulting model with testing to be immediately followed determines its accuracy and precision along with other important metrics. Finally, the entire working of the model is carried out in with the help of a real-time simulation to showcase its functionality in being able to identify the various types of road conditions and in return automatically change its driving mode.

## 3. Implementation

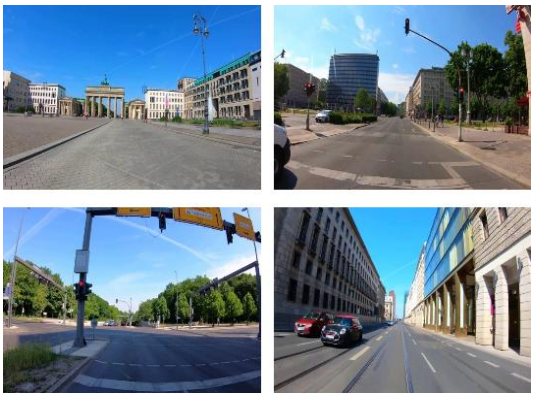
### *3.1 Dataset Collection*

The dataset collection is carried out and based around the classification of various road conditions that is identified prior to creating the neural network model. As mentioned previously the various road classes include Plain Roads, Wet/Slurry Roads, Snowy/Icy Roads. Roughly more than 1000 images have been obtained from downloaded dashcam footage that is readily available on the internet with the help of a frame-by-frame extraction algorithm.

The following are the factors related to this step:

- Number of Dashcam Footages: 3
- Duration of Dashcam Footages: Between 20 minutes to over 1 hour
- Images collected for Plain Roads: 1099
- Images collected for Wet/Slurry Roads: 1385
- Images collected for Snowy/Icy Roads:1035
- Total number of Images: 3519
- Count: 35 (In the case for Wet/Slurry Roads, changes for each class)

3 dashcam footages, each corresponding to a different class of road condition was utilized. An important observation to take note of in this step is that the images obtained are during the day when the lighting conditions are optimum which helps in identifying key elements and characteristics of each road conditions much better as compared to during the night or in instances where the lighting conditions are not favorable. At this stage there is no constraint applied towards the size or the resolution of the images collected as it is obtained directly from the videos of the dashcam footage.



**a).** Plain/Clear Roads

**b).** Wet/Slurry Roads

**c).** Snowy/Icy Roads

**Figure 1. Various classes of roads**

*3.2 Pre-Processing*

The images that have been obtained consist of a lot of unnecessary information, with the main aim being to view the condition of the road and not much more, pre-processing techniques such as image cropping and re-sizing are carried out. This is also implemented with the help of a pre-processing algorithm.

The constraints include:

- Image Crop: [350:720, 0:1280]
- Resize: (224,224)

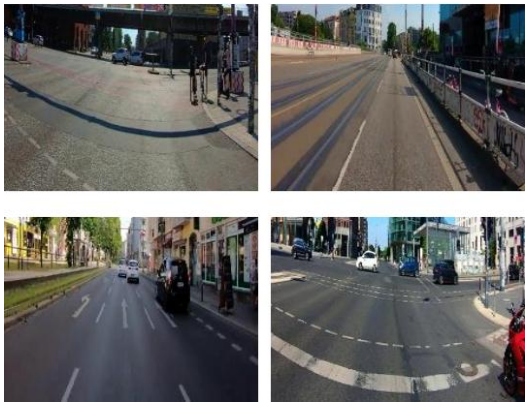


Figure 2. Examples of Pre-Processed Images for Plain/Clear Roads

This helps the model to identify certain features and train much more efficiently leading to accurate predictions and results. The resulting images obtained from the algorithm are well labelled and once again stored separately from the images obtained previously for ease of access and recall in the steps that are to be accompanied. These images will be referred to as the Model's Dataset.

### *3.3 Transfer Learning and Fine-Tuning*

In this step, Python along with the help of miniconda and Jupyter notebook will be used to aid the creation of the neural network. The necessary libraries including the likes of NumPy, Pandas, TensorFlow, Keras and more which are key to the model being created are already available. Following this the model's dataset collected previously will undergo further processing before being passed onto the testing phase. An example would be the use of ImageDataGenerator, this class provides a quick and simple solution in enhancing the existing images and offers a variety of augmentation techniques like flips, rotation, shifts and much more. This benefits the training process by providing new variations leading to better results.

The factors with the help of ImageDataGenerator include:

- Shear Range: 10
- Zoom Range: 0.4
- Horizontal Flip: True

Example of parameters that are also present in this stage include:

- Identifying the Percentage of Images Used for Training and Testing: 0.8 and 0.2 (80% and 20% respectively)
- Batch Size: 128
- Target Size:(224,224)

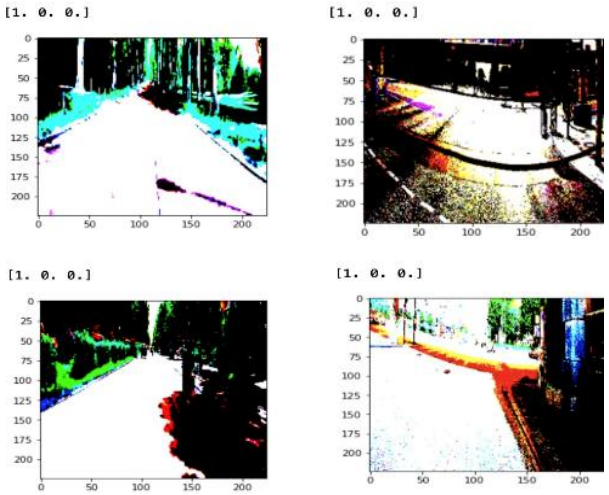



Figure 3. Examples of Images After Processing

80% of the images from the model's dataset will be used during the training phase whereas the other 20% (703 images) will be used for testing purposes which indicates the performance of the resulting model obtained from the testing.

#### *3.3.1 Transfer Learning*

The ability to call upon ResNet50's model and architecture is achieved by transfer learning with the help of Keras. The ResNet50 already has pre-trained weights from ImageNet, which are a large database that contains millions of images. There are many advanced models that exist and is stored within Keras that can be recalled anytime with ease and for various purposes.

### *3.3.2 Fine Tuning*

Fine-Tuning is another technique that allows layers to be "frozen", this means that the layers as well as some of the parameters already present within the pre-trained ResNet50 architecture are not varied or altered. However, with the addition of certain technical parameters and new layers, the pre-trained model is tailored to accommodate for the new model that is being created to identify various road conditions.

The technical parameters include:

- Layer Trainable: False
- Dense layers: 3
- Activation function: ReLU
- Pooling type: Average Pooling
- Softmax function

Providing layer trainable as false initiates freezing of all the weights. The dense output layers which are 3 stems from the classes of road conditions that are present. The activation function known as ReLU is an important parameter involved in this step. This is used in comparison to other activation functions because it is much simpler, faster, seems to function well and converges more quickly during training. The Softmax function that is used acts as an activation function in the output layer, whereas average pooling is one of the various types of pooling operation that can calculate the average value for several patches of a feature map, a feature map is the result of applying various number of filters to an input image. This results in a downsized or a down sampled feature map. Fine-tuning and transfer learning work in tandem which results in not having to create a new neural network from the base up as it is very difficult and a time-consuming process with a lot of small technical aspects integrated.

| | | | |
|---|---|---|---|
| stage4_unit3_conv3 (Conv2D) | (None, 7, 7, 2048) | 1048576 | stage4_unit3_relu3[0][0] |
| add_15 (Add) | (None, 7, 7, 2048) | 0 | stage4_unit3_conv3[0][0]<br>add_14[0][0] |
| bn1 (BatchNormalization) | (None, 7, 7, 2048) | 8192 | add_15[0][0] |
| relu1 (Activation) | (None, 7, 7, 2048) | 0 | bn1[0][0] |
| global_average_pooling2d (Globa | (None, 2048) | 0 | relu1[0][0] |
| dense (Dense) | (None, 128) | 262272 | global_average_pooling2d[0][0 |
| dense_1 (Dense) | (None, 3) | 387 | dense[0][0] |

Total params: 23,808,716
Trainable params: 262,659
Non-trainable params: 23,546,057

Figure 4. Creation of ResNet50 Architecture and Metrics

*3.4 Training*

Once the new model has been created, training can then proceed, the dataset that was set aside (0.8 or 80%) for training will now be called upon and integrated. The crucial parameter in this step is the number of epochs run and the type of optimizer used. While training occurs certain metrics such as the validation loss and accuracy can be viewed simultaneously with the addition of plots. This estimates how well the model can train, update and self-learn.
Some of the important parameters involved in this step include:

- Number of epochs: 5 epochs
- Type of optimizer: Adam

The number of epochs tend to also update the existing trainable weights the same number of times. This can lead to the boundary of the model shifting from underfitting to optimal and finally to overfitting. The optimizer Adam is utilized because it is generally superior compared to several other optimization algorithms. Adam has a comparatively much faster computation time and few parameters requirement for tuning. The resulting model is saved in a .h5 format and will be used for testing purposes and subsequently to carry out real-time simulations.

*3.5 Testing*

Testing utilizes the remaining portion (0.2 or 20%) also referred to as the validation dataset to study the saved trained model to determine its functionality as well as its accuracy in identifying the various road conditions that were initially set out in the beginning. The findings from this step provide essential metrics which include accuracy, precision and also F1 score. This is further studied during the results and discussion section. This step is quite important as it allows us to study the resulting information and decide if there are any changes or improvements that can and must be proposed.

## 4. Inference

A new dataset which is also referred to as the “Test Dataset” consisting of relatively fewer images with a mixed combination of different classes of road conditions is created and used in order to carry out a real-time simulation with the model created thus far. The model is tested with the help of 2 consecutive simulations.

*4.1 Simulation 1*

The model is fed with the test dataset to first view whether it is able to recognize the specific road conditions present within the new dataset. This initial indication shows the capability as well as the functionality of the model. For ease and aesthetic purposes, the predictive percentages will be provided along each road condition. The existing classes are rebranded to:

- Plain Roads= “Plain”
- Wet/Slurry Roads= “Wet”
- Snowy/Icy Roads= “Snow”

Predictive percentages are a method used to show and visualize how the model classifies and interprets the information present within the images of the test dataset.

*4.2 Simulation 2*

A change in the vehicle’s driving mode is then observed with the resulting classification of road

condition obtained from the first simulation to accommodate for it. The following are the driving modes that relate to each class and will be presented following the first simulation:

- Plain Roads= “NORMAL MODE”
- Wet/Slurry Roads= “WET MODE”
- Snowy/Icy Roads= “SNOW MODE”

## 5. Results

During the testing phase, the number of epochs ran was 5 and the resulting information is shown in table 1. It is evident that the loss at the beginning was quite high with 95.49%, however, reduced to 25.64% by the end of the training process. This shows the model created was able to run through the images provided in the training dataset and learn quite efficiently. This consequently resulted in a final accuracy result of 91.67%. From the metrics the model initially does not exhibit any signs of overfitting. Overfitting would have resulted in much higher prediction results whereas the results obtained show a steady increase in both the accuracy as well as validation accuracy with each epoch and levelling off. On the other hand, underfitting can also not be observed due to the metrics obtained.

Table 1. Testing results

| Epoch | Time Taken (s) | Loss (%) | Accuracy (%) | Val. Loss (%) | Val. Accuracy (%) |
|---|---|---|---|---|---|
| 1 | 0 | 95.49 | 64.14 | - | - |
| | 86 | 95.49 | 64.14 | 51.72 | 74.96 |
| 2 | 84 | 46.14 | 84.29 | 31.06 | 92.32 |
| 3 | 85 | 37.89 | 86.96 | 23.84 | 93.74 |
| 4 | 86 | 30.06 | 90.03 | 19.89 | 94.31 |
| 5 | 85 | 25.64 | 91.67 | 16.36 | 95.31 |

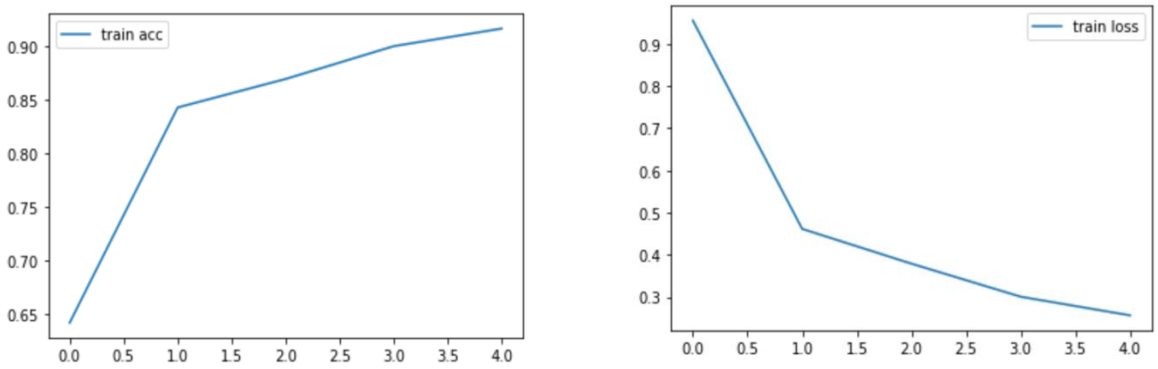



Figure 5. Training Accuracy and Loss Plots Respectively

The plot in figure 5. relay the results from the table above showing a steady increase and decrease in accuracy and loss respectively. The figures obtained after testing indicate that the model created is quite accurate and precise with a score of 95.31% and 95.48% respectively. A recall score of 95.12% is recorded showing that the model can correctly identify and detect positive samples. Whereas the F1 score obtained is 95.26% which is harmonic mean of both the precision and recall scores.

```
Accuracy: 0.953058
Precision: 0.954754
Recall: 0.951228
F1 score: 0.952614
```

Figure 6. Metrics Obtained from Testing

*5.1 First Simulation: Detection of Road Condition*

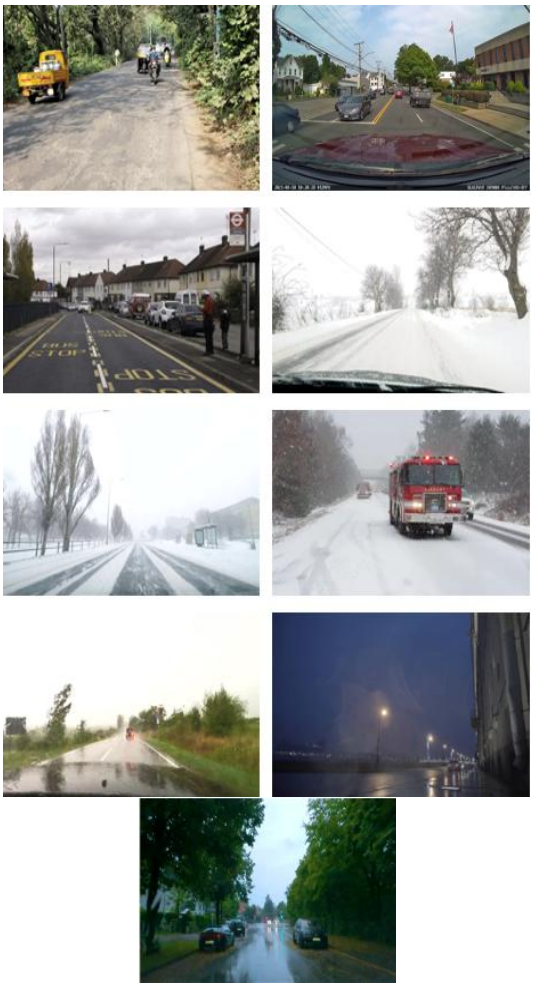


Figure 7. Test Dataset for Simulation

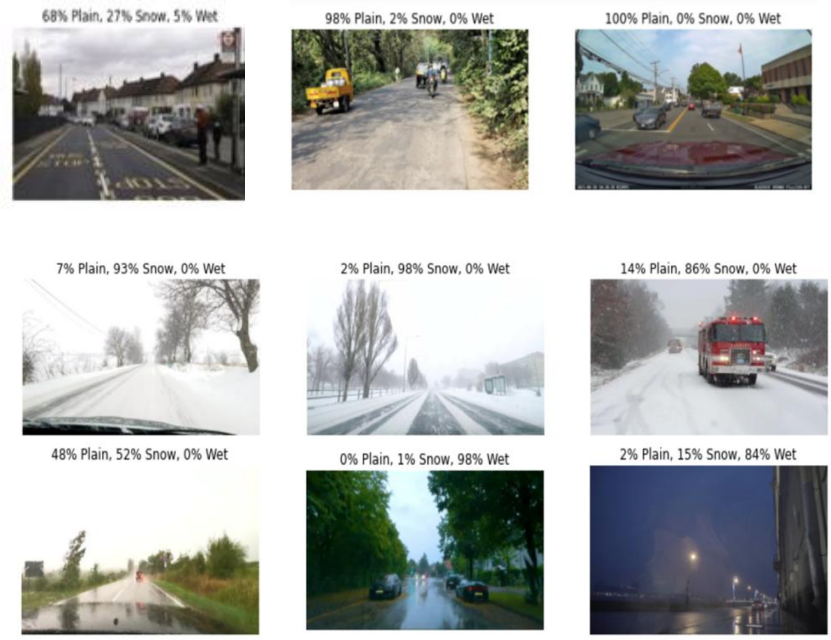



Figure 8. Identification of Respective Road Condition with Predictive Percentages

As seen in Figure 10. the model was able to provide predictive percentages on each of the images that

was provided in the test dataset. This helps the user observe how the model works and can interpret various sorts of information.


Figure 9. Predictive Percentages for Snowy/Icy Road

An image characterizing snowy road conditions as seen in figure 11. shows that the highest percentage, 98%, relates to the “Snow” class whereas a mere 2% and 0% were assigned to the “Plain” and “Wet” class respectively. Overall, out of the 9 images that were provided to the model, 8 of them resulted in the correct classification. For one of the images that contained a wet/slurry road condition, the resulting prediction was in favour to a 52% “Snow” class instead of identifying a wet road as seen in figure 10.


Figure 10. Incorrect Classification for Wet/Slurry Roads

This evidently shows that there is in fact a slight overfitting within the model, showing a direct contradiction to the observation made in the testing phase. However, this problem poses no immediate and actual threat to the working of the model as an increase in images collected in the actual dataset for training should be a guaranteed fix. With the confirmation that the model is indeed working, the second simulation can proceed.

*5.2 Second Simulation: Driving Mode*

Figure 11. shows the actual change in driving mode with each one of the images provided to it. With the help of predictive percentages from the previous simulation, the driving mode automatically redesigns itself to switch to a driving mode class that relates to the road condition class. This was displayed by using a phrase that is manually entered by-hand: DRIVING MODE= “class”, where class responds to the driving mode assigned to their respective road condition. Figure 14. shows the perfect example of a working model. The consensus behind both simulations relates to first identifying the class of road condition with predictive percentages and the highest percentage will result in the respective driving mode accommodating the previous prediction. With this the creation of a model that is successfully able to detect various road conditions and in turn respond with a change in driving mode has been carried out and completed entirely within its software capabilities to aid autonomous vehicles.


Figure 11. Change of Autonomous Vehicle Driving-Mode

## 6. Conclusion

This paper was successfully concluded with it having met all the initially requirements and objectives, thus being able to create a functional and working system to aid autonomous vehicles in identifying various road conditions and resulting in the change of its driving mode automatically to accommodate said road condition without the assistance of any human beings. The use of neural network provided an excellent gateway throughout this paper. Several neural networks were available, however convolutional neural networks, specifically ResNet proved to be the best out of them all. Bypassing the vanishing gradient problem allowed greater accuracy during the training phase and allowed integration of multiple layers to further improve its classification prowess. Frame-by-frame extraction and pre-processing algorithms help obtain datasets much faster and focus on the information that is intended and necessary. The use of data augmentation techniques made possible with ImageDataGenerator allows different variation of images during the training phase since convolutional neural networks are invariant to changes in positional orientations and does also help in preventing overfitting problems. Other parameters that were involved, including the likes of activation function, type of pooling and optimizer used, all contributed to optimizing the working and the creation of the model. However, mentioning overfitting, the results obtained during inference did provide a single miss diagnosis on a particular image. This is yet no cause for concern since neural networks can improve themselves and become more accurate when extra sets of datasets or images are provided, guiding them to learn much better. The use of transfer learning and fine-tuning makes the creation of such complex networks much easier. The addition of libraries such as TensorFlow and Keras also chip into this allowing pre-trained models to be recalled and implemented into other projects for different and various domains. The creation of the model with the help of python and the other relevant software also helped provide ease and comfort as it was user-friendly and provided necessary information whenever a problem arose. Overall, obtaining a 95.3% and a 95.5% in terms of accuracy and precision respectively of the model with addition to both simulations working out quite well indicates a successful classification model being created.

**References**


1. Ajit, Arohan, Koustav Acharya, and Abhishek Samanta 2020 A review of convolutional neural networks. 2020 international conference on emerging trends in information technology and engineering (ic-ETITE). 1-5
2. Cengiz, Enes, et al. 2019 Pedestrian and Vehicles Detection with ResNet in Aerial Images. 4th. International Symposium on Innovative Approaches in Engineering and Natural Sciences, Samsun, Turkey
3. Hossain, Md Belal, et al. 2022 Transfer learning with fine-tuned deep CNN ResNet50 model for classifying COVID-19 from chest X-ray images. Informatics in Medicine Unlocked 30: 100916
4. J. A. Calderon-martinez and P. Campoy-Cervera 2006 An Application of Convolutional Neural Networks for Automatic Inspection. 2006 IEEE Conference on Cybernetics and Intelligent Systems. 1-6
5. Jung, Heechul, et al. 2017 ResNet-based vehicle classification and localization in traffic surveillance systems. Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 61-67
6. Kehtarnavaz, N., and W. Sohn 1991 Steering control of autonomous vehicles by neural networks. 1991 American Control Conference. 3096-3101
7. Khan, Md Nasim, and Mohamed M. Ahmed 2021 Weather and surface condition detection based on road-side webcams: Application of pre-trained convolutional neural network. International Journal of Transportation Science and Technology. 11(3): 468-463
8. L. Jiao et al. 2019 A Survey of Deep Learning-Based Object Detection. IEEE Access. (7): 128837-128868
9. Narin, Ali, Ceren Kaya, and Ziynet Pamuk 2021 Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks. Pattern Analysis and Applications. 24(3): 207-1220
10. Rajeshwari, M., and C. H. MallikarjunaRao 2021 Detecting anomalous road traffic conditions using VGG19 CNN Model. E3S Web of Conferences. (309)
11. Sadad, Tariq, et al. 2021 Brain tumor detection and multi-classification using advanced deep learning techniques. Microscopy Research and Technique. 84(6): 1296-1308
12. Saini, Sanjay, et al. 2017 An efficient vision-based traffic light detection and state recognition for autonomous vehicles. 2017 IEEE Intelligent Vehicles Symposium (IV). 606-611
13. W. Rawat and Z. Wang. 2017 Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. Neural Computation. 29(9): 2352-2449
14. Wu, Zifeng, Chunhua Shen, and Anton Van Den Hengel 2019 Wider or deeper: Revisiting the resnet model for visual recognition. Pattern Recognition. (90): 119-133
15. Z. Zhang 2021 ResNet-Based Model for Autonomous Vehicles Trajectory Prediction. 2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE). 565-568
16. Z. Zhao, P. Zheng, S. Xu and X. Wu. 2019 Object Detection with Deep Learning: A Review. IEEE Transactions on Neural Networks and Learning System. 30(11): 3212-3232
17. Z. Zuo, K. Yu, Q. Zhou, X. Wang and T. Li. 2017 Traffic Signs Detection Based on Faster R-CNN. 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW). 286-288.