DrillBit

The Report is Generated by DrillBit Plagiarism Detection Software

Submission Information

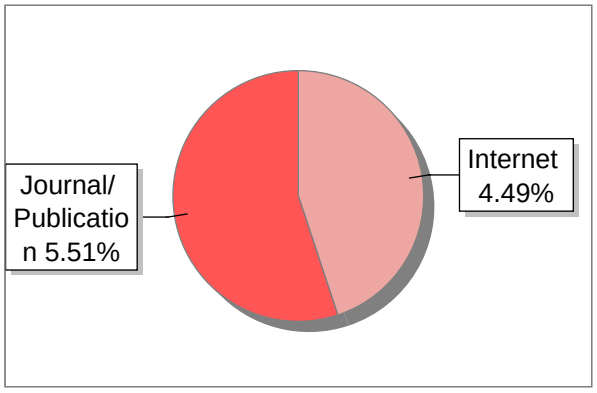
Author Name	aakanksha
Title	insurance
Paper/Submission ID	746691
Submission Date	2023-05-20 02:39:36
Total Pages	31
Document type	Research Paper

Result Information

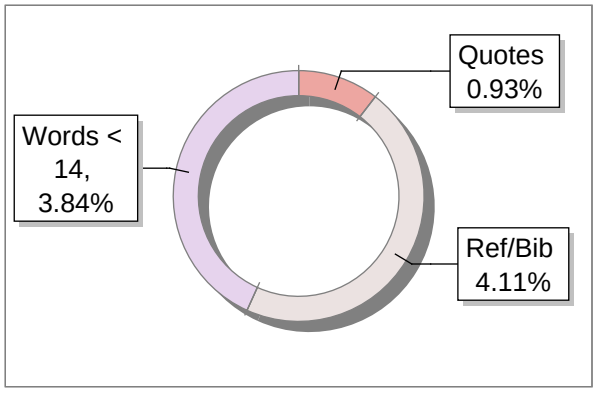
Similarity **10 %**

1 10 20 30 40 50 60 70 80 90

Sources Type



Report Content



Exclude Information

Quotes	Not Excluded
References/Bibliography	Not Excluded
Sources: Less than 14 Words Similarity	Not Excluded
Excluded Source	**0 %**
Excluded Phrases	Not Excluded

A Unique QR Code use to View/Download/Share Pdf File

DrillBit

DrillBit Similarity Report

10	28	A
SIMILARITY %	MATCHED SOURCES	GRADE

A-Satisfactory (0-10%)
B-Upgrade (11-40%)
C-Poor (41-60%)
D-Unacceptable (61-100%)

LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1	Multi-Head CNN-RNN for Multi-Time Series Anomaly Detection An industrial case s by Canizo-2019	1	Publication
2	www.dx.doi.org	1	Publication
3	Deep Learning on Windows Building Deep Learning Computer Vision Systems on Micr by Thimir-2021	1	Publication
4	github.com	<1	Internet Data
5	www.cisco.com	<1	Internet Data
6	machinelearningmastery.com	<1	Internet Data
7	www.jairm.org	<1	Publication
8	dokumen.pub	<1	Internet Data
9	Thesis Submitted to Shodhganga Repository	<1	Publication
10	moam.info	<1	Internet Data
11	1library.co	<1	Internet Data
12	ijircce.com	<1	Publication
13	moam.info	<1	Internet Data

14	coek.info	<1	Internet Data
15	scholar.sun.ac.za	<1	Internet Data
16	TRANSPORT AND MIXING OF by Montes-2016	<1	Publication
17	www.sciencegate.app	<1	Internet Data
18	Blind Signal Separation for Medical Data Recording Using Self-Organizing Neural by Sahroni-2015	<1	Publication
19	bmcgenomics.biomedcentral.com	<1	Internet Data
20	IEEE 2017 National Information Technology Conference (NITC)-Colom, by Fernando, Mahendra - 2017	<1	Publication
21	Large-scale urban functional zone mapping by integrating remote sensing images a by Du-2020	<1	Publication
22	Climate Change Politics by Bernauer-2013	<1	Publication
23	Combination of Spatial and Frequency Domains for Floating Object Detection on Co by Sun-2019	<1	Publication
24	easychair.org	<1	Internet Data
25	qdoc.tips	<1	Internet Data
26	stackoverflow.com	<1	Internet Data
27	www.indianhealthyrecipes.com	<1	Internet Data
28	www.ualberta.ca	<1	Internet Data

Chapter 1

INTRODUCTION

Car insurance companies waste millions of dollars annually, due to claims leakage. Claim processing is long and tedious task and requires a person to remotely verify damage. It takes a long time to receive payments from insurance company. AI-based solutions such as automated car damage detection models can help in remotely identify damage without human intervention. Deep learning can help to improve the assessment process and ensure faster disbursal of claims.

Data is essential and significantly affects both organizations and people. Data storage, computations, and transactions are essential to almost every firm. The world of insurance is highly regulated, which often leads to delays in processing an insurance claim.

With AI, car damage detection and remote assessments are automated and the manual intervention is drastically reduced. It will eliminate the need for human interference for identifying damage.

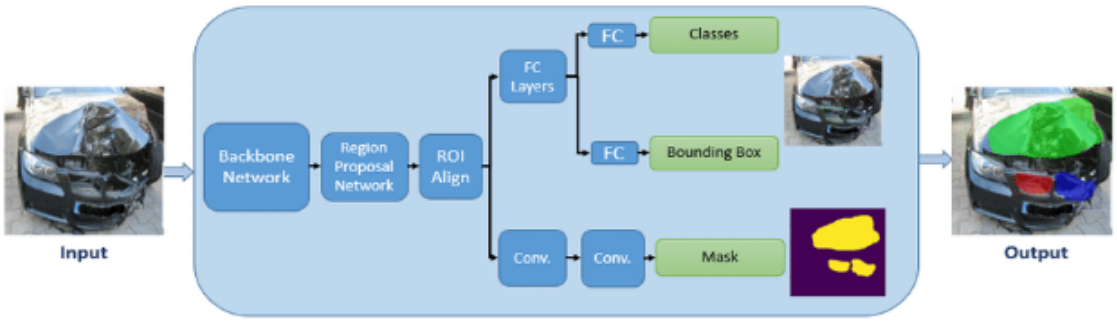


Fig 1.1: The Process of Damage Detection

As can be seen in Fig 1.1, The solution helps to automate the process of assessing car damage right at the accident scene. With a simple interface, users enjoy the possibility to upload photos of a damaged vehicle, and machine learning algorithms do all the magic. This system can be employed by insurance companies, car rental services and body shops.

This very solution comes helpful to those industries and organizations that deal with massive arrays of textual data on a daily basis.

The solution enhances extracting and analysing meaningful information from PDF-based documents.

Our main goal is to analyze damage from multiple angles. It classifies live images from a camera.The model uses an advanced edge deployment mechanism that eliminates the need for human interference for identifying damage estimates for cars. Model should have high accuracy around 95-97%.

Table 1.1 Summary of the Approaches

Approach	Advantages	Disadvantages
CNN	Very high accuracy inthe image recognition.	Orientation and position of the object isnot encoded.
ResNet	Can provide network to boostfor getting through vanishing gradients.	These deeper networks require weeks of training, making it pratically infeasible forreal-world applications
DenseNet	Reduces interdependence between layers.	Each layer maps are spliced with previous layers.

1.1Problem Statement

To enable insurance companies to assess the damages remotely with the help of AI-based car damage detection. With AI, car damage detection and remote assessments are automated and the manual intervention is drastically reduced. It will eliminate the need for human interference for identifying damage estimates for cars.

1.2 Objectives

Artificial Intelligence and Automation are key players in transforming businesses by saving time and cost and reducing manual work and providing efficient recommendations. They will also help address "moon shot" societal challenges.

At the same time, these technologies will transform the nature of work and the workplace itself. Machines are much better at performing complex tasks in few seconds that humans, and even perform some tasks that go beyond what humans can do.

Our main goal is to offer a practical means of detecting car damage using AI. This can be accomplished without needing to decode the dataset by executing processed images directly on the output that we obtain after applying a machine learning algorithm to a specific dataset.

1.3 Relevance of the Problem

To enable computation on processed image samples, Artificial Intelligence Automation methods are primarily used. Thus, data can remain private throughout processing, enabling beneficial tasks to be carried out with data residing in unreliable contexts.

The method of image processing is used to do some processes on a picture like an image enhancement or to remove some functional data from the image. Image processing is one kind of signal processing, where the input is a picture, as well as the output, are features or characteristics allied with the image.

1.4 Software Development Tools

We have used the tools for application is:

- Python
- Jupyter Notebook
- Google Colab
- Kaggle

There were certain libraries used in Python to complete the task:

- Seaborn
- Keras
- Pickle
- Pandas
- Numpy
- Vgg16

1.5 Schedule

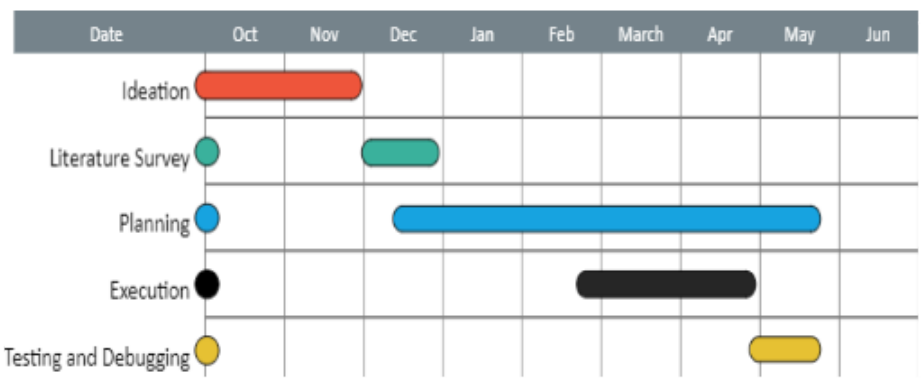


Fig 1.2: Gantt chart

1. The phase of ideation lasted from October to the end of November.Literature Survey was done in the month of December.
2. Planning of the application started in mid-December and went till the end of March. Execution of the project started in February-end and went on till the month of April.
3. Testing and debugging went on for the month of April.

1.6 Chapter Wise Summary

Chapter 1 consists of the introduction and a general overview of this report.

Chapter 2 consists of the Literature survey performed while analysing the best approach that can be considered to solve this problem.

Chapter 3 consists of the Proposed Architecture and System Design which we had decided while developing the application.

Chapter 4 consists of the actual implementation details which exist in the final product that covers the scope of the problem statement at hand.

Chapter 5 consists of the Results achieved from the application and certain aspects of shortcomings as well as how these results can be used in future models especially when expanding the domain of the problem statement.

CHAPTER 2

LITERATURE SURVEY

2.1 Overview

Various sources referred to include papers published through IEEE Access, IEEE Xplore, Conference series: Journal of Physics, ICCECE (International Conference on Consumer Electronics and Computer Engineering), ICECIT (International Conference on Electronics, Communication and Information Technology), ACOMP (International Conference on Advanced Computing and Applications), ISI World Statistics Congress Hindawi, Society for Science and Education (United Kingdom).

Numerous academics put forth various strategies for applying Artificial Intelligence and Deep Learning-based damage classification and Detection. We will examine the study done by many researchers in this topic in thispart.

The method and approach used in Paper [1] to detect damage from the medical dataset for calculating the insurance and apply the machine learning algorithm to the processed images dataset are presented. AI Automation can be effectively utilized to protect the process of images in the cloud since it allows processing to happen while the image is analyzed. To give a more complete view, many methods were used, including CNN and other deep learning-based methods.

2.2 Insurance Calculation using NLP

The main objective of paper [2] is about the insurance industry, NLP can be used to analyze customer inquiries, claims, and other forms of written or spoken communication in order to better understand the needs and concerns of customers. This can help insurance companies to more accurately calculate premiums, assess risk, and offer personalized insurance products and services to customers.

For example, an insurance company might use NLP algorithms to analyze customer inquiries submitted through an online form or over the phone. By analyzing the language and context of these inquiries, the company can identify specific issues or concerns that the customer has, such

as the type of coverage they are interested in or any specific risks or exposures they are concerned about. This information can then be used to tailor the insurance products and services offered to the customer, and to calculate a more accurate premium for their specific needs.

In addition to analyzing customer inquiries, NLP can also be used to analyze claims and other forms of written communication in order to extract relevant information and automate certain tasks, such as evaluating the validity of a claim or determining the appropriate course of action. This can help to improve the efficiency and accuracy of the insurance claims process and can lead to better outcomes for both the insurance company and its customers.

Natural language processing (NLP) is a field of artificial intelligence (AI) that focuses on enabling computers to understand, interpret, and generate human language. NLP techniques can be used to analyze text, speech, and other forms of linguistic data in order to extract meaning and extract useful information.

2.3 VGG16 Neural Network Architecture

The Visual Geometry Group at the University of Oxford has suggested the deep convolutional neural network architecture VGG16 for picture classification. In the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), it was among the top-performing models.

16 convolutional layers make up the VGG16 architecture, which is followed by 3 fully linked layers. Each of the five blocks that make up the convolutional layers has several 3x3 convolutional layers, which are then followed by a max pooling layer. As we move further into the network, the number of filters in each block rises, from 64 in the first block to 512 in the final block.

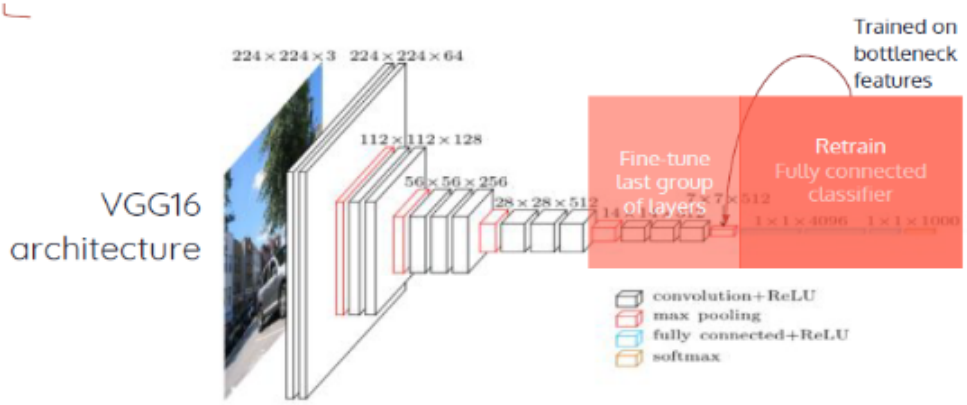


Fig 2.1 VGG16 Architecture

The VGG16 model has about 138 million parameters, making it a large and computationally expensive network. However, it has shown excellent performance on a wide range of image classification tasks, and its architecture has served as a basis for many other popular deep learning models.

2.4 Machine Learning Techniques

The creation of a model that can perform the following operations on processed data is described in Paper [7].

Training paradigm - Networks of convolutional neurons (CNNs) - These are a specific kind of artificial neural network made for processing data with a grid-like layout, such an image. They are widely utilised in a range of applications, such as object identification, face recognition, and picture production. They are particularly good at image categorization and recognition tasks.

CNNs are made up of a number of interconnected layers, each of which performs a particular function in the data analysis and interpretation process. Convolutional layers are the initial layers in a CNN and are in charge of extracting features from the input data.

These characteristics are then transferred through other layers, referred to as "fully connected layers," which make use of the features to anticipate the future or make judgements based on the data at hand.

The focus of Paper [9] is on how CNNs may "learn" from data, which is one of their fundamental characteristics. A CNN may be trained by exposing it to a large dataset of labelled pictures through a process known as "training," which teaches it to spot trends and features in the data that are pertinent to a particular task. This enables CNN to predict or decide more accurately when faced with fresh, previously undiscovered facts.

2.5 Classification of Input Data

The stages that are commonly used for classifying raw image collections are as follows:

2.5.1 Data Gathering

Gather a lot of unprocessed photos that correspond to the classes you want to classify.

2.5.2 Data Pre-Processing:

To prepare the data for the machine learning model's efficient processing, clean the data, normalise the pixel values, and resize the photos to a uniform size.

2.5.3 Data Splitting

Splitting the data into training, validation, and testing sets is known as data splitting. The validation set is used to fine-tune the model's hyperparameters, the testing set is used to assess the model's performance, and the training set is used to train the model.

2.5.4 Model Selection

Select a machine learning model that is appropriate for the categorization task. Convolutional neural networks are some well-liked image classification models.

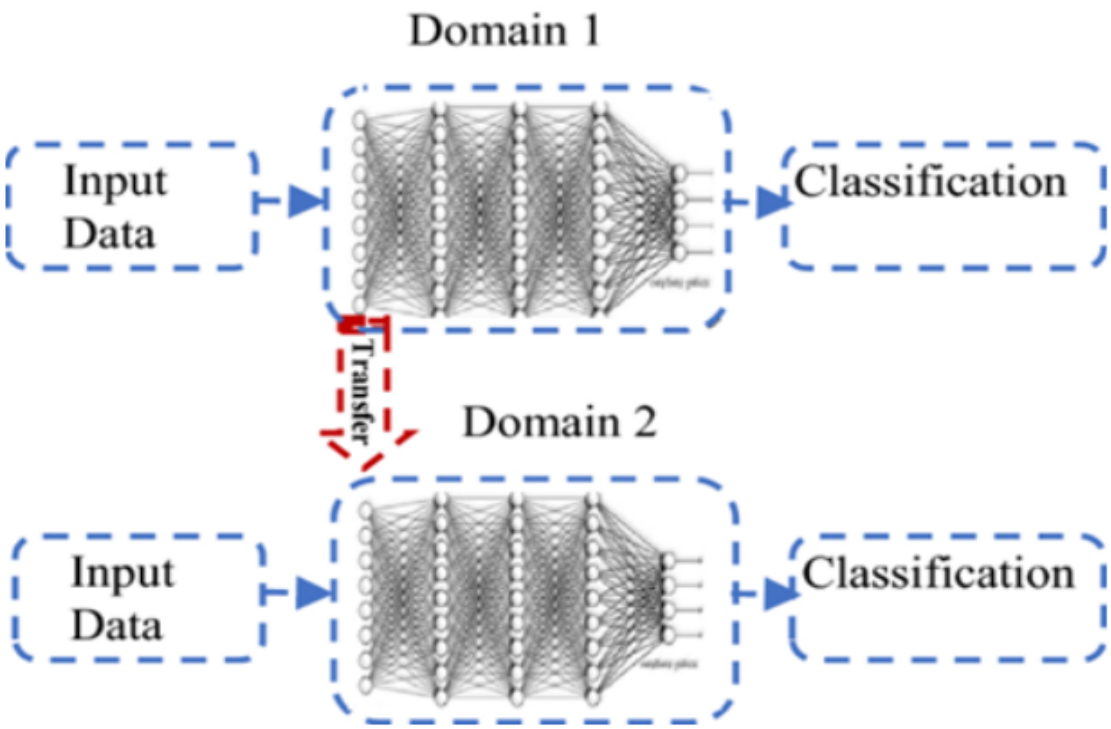


Fig 2.2 Classification of Input Data

2.7 Convolutional Neural Networks (CNNs)

For image classification applications, convolutional neural networks (CNNs) are a common variety of neural network. An overview of the procedures for employing CNNs on a picture dataset is provided below:

Gather the data: Collect and pre-process the picture dataset, dividing it into training, validation, and testing sets, as well as resizing the photos and normalising the pixel values. Describe the architecture of CNN: Create the CNN architecture, which usually comprises of a number of convolutional layers, pooling layers, and fully linked layers.

Compile the model: When compiling the model, be sure to include the evaluation metrics, optimizer, and loss function. The optimizer modifies the model parameters to minimise the loss function during training, and the loss function gauges how well the model is functioning.

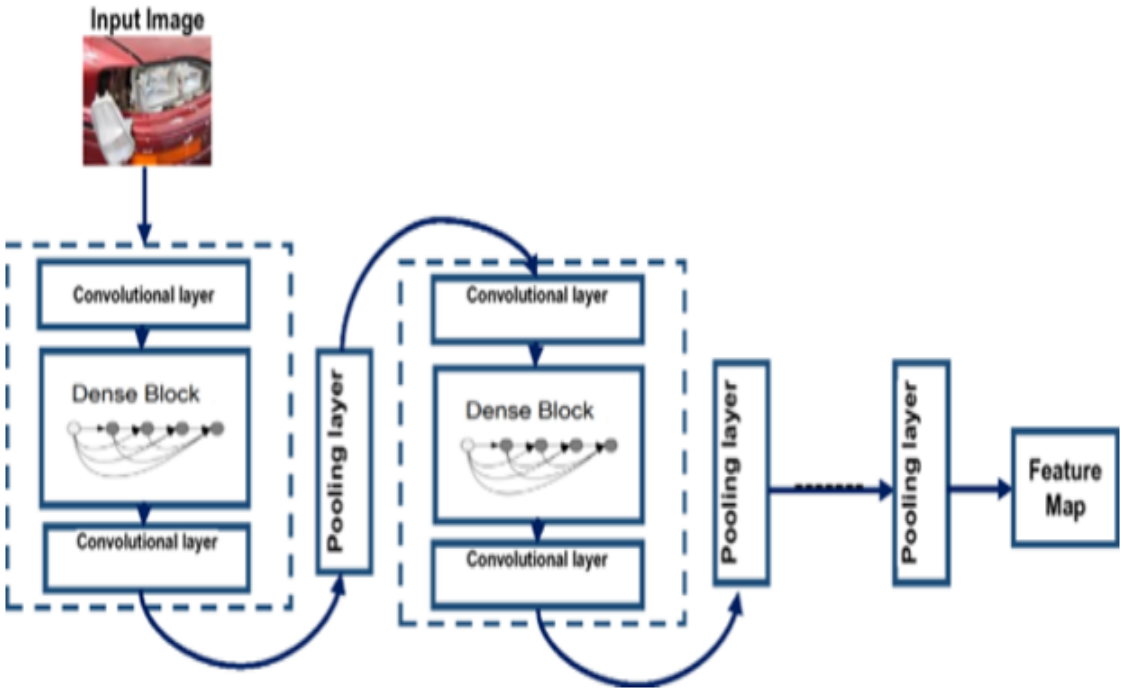


Fig 2.3 Convolutional Neural Networks (CNNs)

2.8 Residual Network (ResNet)

For image classification applications, convolutional neural networks (CNNs) are a common variety of neural network. An overview of the procedures for employing CNNs on a picture dataset is provided below:

Gather the data: Collect and pre-process the picture dataset, dividing it into training, validation, and testing sets, as well as resizing the photos and normalising the pixel values. Describes the architecture of CNN: Create the CNN architecture, which usually comprises of a number of convolutional layers, pooling layers, and fully linked layers.

Compile the model: When compiling the model, be sure to include the evaluation metrics, optimizer, and loss function. The optimizer modifies the model parameters to minimise the loss function during training, and the loss function gauges how well the model is functioning.

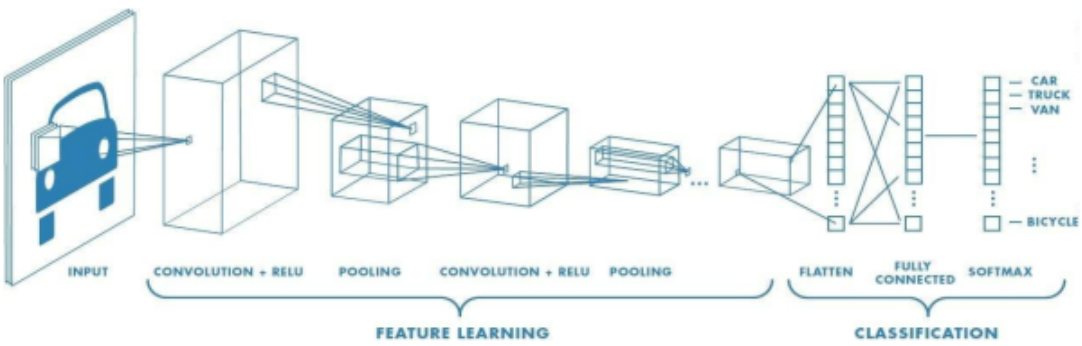


Fig 2.4 Residual Network (ResNet)

2.9 Accuracy Analysis of all the Models

Depending on the particular problem, dataset, and model architecture being employed, the accuracy of machine learning models might vary significantly. Here are a few illustrations of typical accuracy ranges for several machine learning model types:

For straightforward classification or regression tasks, linear models, such as logistic regression or linear regression, are frequently utilised and have an accuracy range of 70–90%. Decision trees: With an accuracy range of 70–90%, decision trees are frequently employed to solve categorization issues.

Support Vector Machines (SVMs): SVMs have an accuracy range of 80–95% and are frequently employed for classification jobs.

Convolutional and recurrent neural networks (CNNs) and other neural networks are potent models that can achieve cutting-edge accuracy on a variety of image and text categorization tasks. Accuracy ranges from 80-90% to over 99%, depending on the difficulty of the problem and the amount of the dataset.

It's crucial to remember that accuracy is only one of the metrics used to assess machine learning models, and that for some tasks, other metrics like precision, recall, F1-score, and area under the curve (AUC) may be more suitable.

The type of data used, the complexity of the model, and the veracity of the labels can all affect how accurate the damage detection model is.

Using sensor data from the machine or structure to train a model to detect changes in the data that are suggestive of damage is one method for damage detection using AIML. Machine learning algorithms like decision trees, support vector machines, or neural networks may be used in conjunction with signal processing, feature extraction, and classification approaches.

2.10 Analysing Algorithm for Auto damage detection

Here's a high-level algorithm for an AI-based system for detecting vehicle damage and estimating insurance:

2.10.1 Pre-Processing

Resize and pre-process the input images to match the required input size of the VGG16 model.

Normalize the pixel values of the images.

2.10.2 Load The VGG16 Model

Load the pre-trained VGG16 model, which has been trained on a large dataset for image classification.

2.10.3 Fine-Tuning (Optional)

If you have a labeled dataset specifically for vehicle damage detection, you can perform fine-tuning on the pre-trained VGG16 model. This involves updating the weights of the model's last few layers to adapt it to the new task.

2.10.4 Training (Optional)

If you don't have a labelled dataset for vehicle damage detection, you can skip this step and proceed with using the pre-trained model directly.

2.10.5 Damage detection

Pass the preprocessed images through the pre-trained VGG16 model to classify if the vehicle is damaged or not.

The output can be binary (damaged/not damaged).

2.10.6 Damage Localization:

If the vehicle is classified as damaged, use techniques such as object detection or semantic segmentation to localize the damaged area in the image.

Object detection models like YOLO, Faster R-CNN, or SSD can be used to detect the bounding box or mask of the damaged area.

Alternatively, semantic segmentation models like U-Net or Mask R-CNN can be used to directly segment the damaged area.

2.10.7 Damage Part Classification

Once the damaged area is localized, use a classification model (e.g., another CNN or a multi-layer perceptron) to classify the specific part of the vehicle that is damaged.

Train the classifier on a labelled dataset containing images of different vehicle parts (e.g., hood, bumper, headlights) with corresponding damage labels.

2.10.8 Insurance Estimation

Based on the classified damage part, use historical data or predefined rules to estimate the insurance cost for repairing or replacing that specific part.

The estimation can be done using regression techniques or by mapping the classified damage part to a predefined cost range.

2.10.9 Post-Processing And Reporting

Aggregate the results of the damage detection, localization, part classification, and insurance estimation.

Generate a comprehensive report indicating whether the vehicle is damaged, the location and type of damage, and the estimated insurance cost.

Optionally, provide visualizations or images highlighting the damaged areas and the corresponding part labels.

CHAPTER 3

PROPOSED ARCHITECTURE AND DESIGN

In this chapter, we highlight the complete architecture that we have proposed for the solution of our problem.

According to the problem statement, we recognized that the insurance claims process can be automated with deep learning model. We identified that damage location and severity are accuracies of 79% and 71% respectively, comparable to human performance. We have trained the model on VGG16 with Keras and Theano to classify damage.

A block diagram of the summarized methodology has been shown in Fig. 3.1.

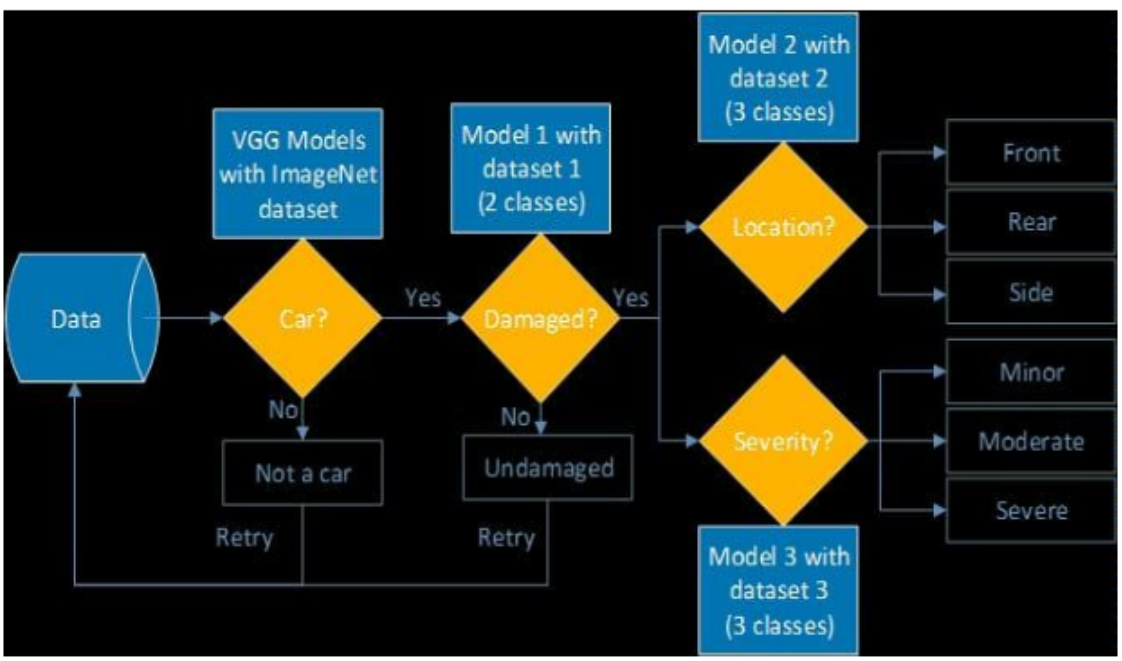


Fig 3.1: Methodology proposed

3.1 Image Classifications

The first phase comprises of the processing of the image received. Fig 3.2 encapsulates the pre-processing of the image where we essentially take an image as input.

Based on the literature survey conducted, we came to the conclusion that a convolutional neural network would suit best to give a feasible solution for the problem statement since the highest accuracies were observed in the research papers using CNN. The CNN model proposed would predict the character given to it as input in the form of an image.

3.2 Layers in CNN

Convolution Layer

The Convolution Operation's goal is to take the input image's high-level characteristics, such edges, and extract them. There is no requirement that ConvNets have just one convolutional layer. Typically, low-level features like edges, colour, gradient direction, etc. are captured by the first ConvLayer. The architecture adjusts to the High-Level characteristics as we add more layers, giving us a network with a comprehensive comprehension of the photos in the dataset.

Pooling Layer

The Convolved Feature's height and width are decreased as a result of the pooling layer. In order to handle the data with less processing resources, dimensionality reduction is used in this procedure. The process of efficiently training the model can be maintained by extracting dominant characteristics that are rotational and positional independent.

Fully Connected Layer

In order to flatten the image into a column vector for our Multi-Level Perceptron after we have finished extracting the necessary features from an image, A feed- forward neural network receives the flattened output, and backpropagation is used for each training iteration. The model can discriminate between dominant and specific low-level features in images and classify them using classification algorithms over a number of epochs.

3.3 Neural Network Architecture

Fig 3.3 shows the VGG neural network architecture that has been designed to train the model. The first Conv layer, is input layer, that accepts the image in grayscale and returns image without any change in its dimensions, that is the neurons present in this layer are just trying to learn the features of the image provided.

The second Conv layer accepts the image of shape (28, 28, 1) and increases the filters while decreasing the first two dimensions in image, hence larger combinations of patterns are being learnt from image in this layer.

The third layer is the MaxPooling layer which decreases the length and breadth to half of what was received, hence regulating training parameters.

The fourth layer is the Dropout layer which exists to prevent any possible cases of overfitting.

The fifth layer is another Conv layer which again increases filters in the image while decreasing the length and breadth and therefore, learning the patterns present in image in greater detail than before, as the filters are now increased to 64.

The sixth layer is another Conv layer with the sole aim of reading patterns from image while decreasing the image size and hence parameters.

The seventh layer MaxPooling, the working of which is the same as the previous MaxPooling layer.

The eighth layer, Dropout layer and have the same function of prevent overfitting as the previous Dropout layer.

The ninth layer, Flatten layer. This layer, exists for flattening input as we now want to flatten the channels of the incoming layers and feed them into the Dense layers. No learning parameters are present in this layer.

The tenth layer, Dense layer, the conventional deep learning layer. Now we will apply the insights and data which we have learned from previous layers for decision making, which is where the Dense layer comes into the picture.

The eleventh layer is another dropout layer which again prevents overfitting like the previous

dropout layers.

The twelfth is another Dense layer with 52 output parameters which tells us which signs, digits, etc would be present in our image.

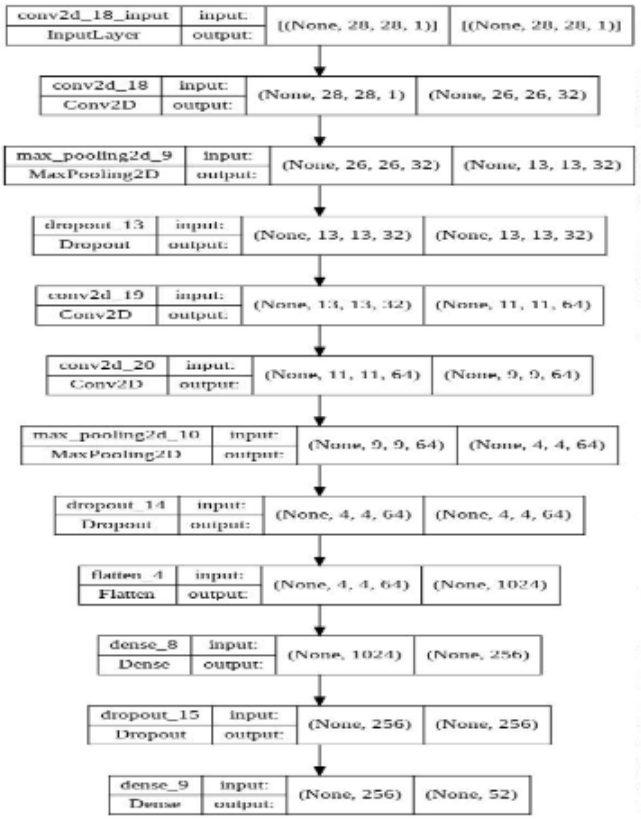


Fig 3.2: Neural Network Architecture

CHAPTER 4

IMPLEMENTATION

In this section we have described the implementation methods followed to build a system that solves the problem statement. The main part of the project which is the code for the application. We can see a few code snippets below which have the details related to a few functionalities in the application. As we have used python, the code snippets are supported by the same. We have included the code snippets of the implementations of functionalities.

4.1 All car images, including whole and damaged

```
from keras.callbacks import ModelCheckpoint, History
from keras import optimizers
from keras.utils.np_utils import to_categorical
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.layers import Convolution2D, MaxPooling2D, ZeroPadding2D
from keras.models import Sequential, load_model
from keras.preprocessing.image import ImageDataGenerator,
array_to_img, img_to_array, load_img
from keras.applications.imagenet_utils import preprocess_input,
decode_predictions
from keras.applications.inception_v3 import InceptionV3
from keras.applications.vgg19 import VGG19
from keras.applications.vgg16 import VGG16
from keras.applications.resnet50 import ResNet50
from keras.utils.data_utils import get_file
import pandas as pd
import numpy as np
import h5py
import os
from sklearn.metrics import classification_report, confusion_matrix
import pickle as pk
import json
```

```
import urllib
from IPython.display import Image, display, clear_output
from collections import Counter, defaultdict


import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline



CLASS_INDEX = None
CLASS_INDEX_PATH='https://s3.amazonaws.com/deep-learning-
models/image-models/imagenet_class_index.json'



def get_predictions(preds, top=5):
    global CLASS_INDEX
    if len(preds.shape) != 2 or preds.shape[1] != 1000:
        raise ValueError('`decode_predictions` expects '
                         'a batch of predictions '
                         '(i.e. a 2D array of shape (samples, 1000)).
'
                         'Found array with shape: ' + str(preds.shape))
    if CLASS_INDEX is None:
        fpath = get_file('imagenet_class_index.json',
                         CLASS_INDEX_PATH,
                         cache_subdir='models')
        CLASS_INDEX = json.load(open(fpath))
    results = []
    for pred in preds:
        top_indices = pred.argsort()[-top:][::-1]
         result = [tuple(CLASS_INDEX[str(i)]) + (pred[i],) for i in
top_indices]
        result.sort(key=lambda x: x[2], reverse=True)
        results.append(result)
```

```
    return results

vgg16 = VGG16(weights='imagenet')
vgg16.save('vgg16.h5')
resnet50 = ResNet50(weights='imagenet')
vgg19 = VGG19(weights='imagenet')
inception = InceptionV3(weights='imagenet')
Image('test.jpg', width=200)
```

Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg19_weights_th_dim_ordering_th_kernels.h5

574717952/574710688 [==============================] - 30s

Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.2/inception_v3_weights_th_dim_ordering_th_kernels.h5

94617600/95119472 [============================>.] - ETA: 0s

Fig 4.1: Generate Test Image

```
def prepare_image(img_path):
    img = load_img(img_path, target_size=(224, 224))
    x = img_to_array(img)
```

```
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    return x
```

4.2 Testing with different Models

```
y = prepare_image('test.jpg')
preds = vgg16.predict(y)
print get_predictions(preds, top=5)
```

[[(u'n02951358', u'canoe', 0.11222993), (u'n03459775', u'grille', 0.079561532), (u'n02974003', u'car_wheel', 0.054067023), (u'n04560804', u'water_jug', 0.036461826), (u'n02795169', u'barrel', 0.03404858)]]

```
z = preprocess_input(x)
preds = resnet.predict(z)
print get_predictions(preds)
```

[[(u'n03459775', u'grille', 0.28048685), (u'n04560804', u'water_jug', 0.26908153), (u'n03950228', u'pitcher', 0.080248252), (u'n04141975', u'scale', 0.071326435), (u'n02974003', u'car_wheel', 0.04589555)]]

```
a = preprocess_input(x)
preds = vgg19.predict(a)
print get_predictions(preds)
```

[[(u'n04493381', u'tub', 0.3188411), (u'n02808440', u'bathtub', 0.19447012), (u'n04447861', u'toilet_seat', 0.071480118), (u'n04049303', u'rain_barrel', 0.06163064), (u'n02951358', u'canoe', 0.050739273)]]

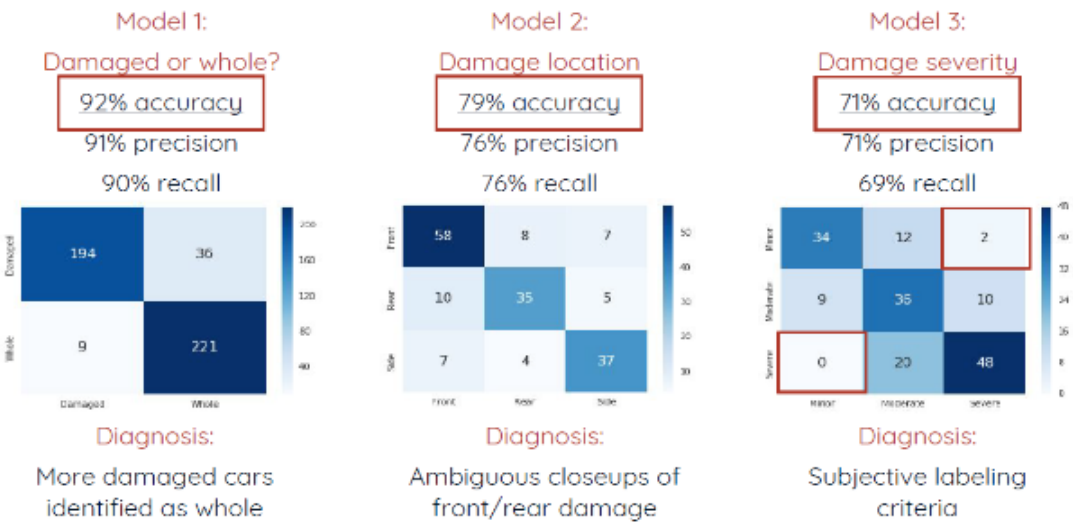


Fig 4.2: Model Component Accuracy

4.3 Using VGG16 as Gate

```
def get_car_categories():
    d = defaultdict(float)
    img_list = os.listdir('data0')
    for i, img_path in enumerate(img_list):
        img = prepare_image('data0/'+img_path)
        out = vgg16.predict(img)
        top = get_predictions(out, top=5)
        for j in top[0]:
            d[j[0:2]] += j[2]
        if i % 50 == 0:
            print i, '/', len(img_list), 'complete'
    return Counter(d)
cat_counter = get_car_categories()

with open('cat_counter.pk', 'wb') as f:
    pk.dump(cat_counter,f,-1)
Load Pickle Point
with open('cat_counter.pk', 'rb') as f:
```

```
    cat_counter = pk.load(f)
cat_list = [k for k, v in cat_counter.most_common()[:50]]

4.4 Evaluating Car Categories

def get_car_categories(cat_list):
    img_list = os.listdir('data0')
    num = 0
    bad_list = []
    for i, img_path in enumerate(img_list):
        img = prepare_image('data0/'+img_path)
        out = vgg16.predict(img)
        top = get_predictions(out, top=5)
        for j in top[0]:
            if j[0:2] in cat_list:
                num += 1
                break # breaks out of for loop if one of top 50
categories is found
            else:
                pass
            bad_list.append(img_path) # appends to "bad list" if
none of the 50 are found
        if i % 100 == 0:
            print i, '/', len(img_list), 'complete'
    bad_list = [k for k, v in Counter(bad_list).iteritems() if v ==
5]
    return num, bad_list

number, bad_list = get_car_categories(cat_list)
```

0 / 1309 complete

50 / 1309 complete

100 / 1309 complete

150 / 1309 complete

200 / 1309 complete
250 / 1309 complete
300 / 1309 complete
350 / 1309 complete
400 / 1309 complete
450 / 1309 complete
500 / 1309 complete
550 / 1309 complete
600 / 1309 complete
650 / 1309 complete
700 / 1309 complete
750 / 1309 complete
800 / 1309 complete
850 / 1309 complete
900 / 1309 complete
950 / 1309 complete
1000 / 1309 complete
1050 / 1309 complete
1100 / 1309 complete
1150 / 1309 complete
1200 / 1309 complete
1250 / 1309 complete
1300 / 1309 complete

```
number2, bad_list2 = car_categories_gate(cat_list2)
```

0 / 1309 complete
100 / 1309 complete
200 / 1309 complete
300 / 1309 complete
400 / 1309 complete
500 / 1309 complete
600 / 1309 complete
700 / 1309 complete
800 / 1309 complete

900 / 1309 complete
1000 / 1309 complete
1100 / 1309 complete
1200 / 1309 complete
1300 / 1309 complete

```
number2, bad_list2 = car_categories_gate(cat_list2)
```

2

4.5 Select top 50 as cutoff for category list

```
def view_images(img_dir, img_list):
    for img in img_list:
        clear_output()
        display(Image(img_dir+img))
        num = raw_input("c to continue, q to quit")
        if num == 'c':
            pass
        else:
            return 'Finished for now.'
view_images('data0/', bad_list)
```

Fig 4.3: View Car Images from Dataset

c to continue, q to quit q

'Finished for now.'

```
def car_categories_gate(image_path, cat_list):
     urllib.urlretrieve(image_path, 'save.jpg')  # or other way to
upload image
    img = prepare_image('save.jpg')
    out = vgg16.predict(img)
    top = get_predictions(out, top=5)
    print "Validating that this is a picture of your car..."
    for j in top[0]:
        if j[0:2] in cat_list:
            print j[0:2]
           return "Validation complete - proceed to damage evaluation"
    return "Are you sure this is a picture of your car? Please take
another picture (try a different angle or lighting) and try again."
car_categories_gate('https://encrypted-
tbn1.gstatic.com/images?q=tbn:ANd9GcSxhKhaSwPgdQkrDegC6sbUALBF9SiW6t
DKg6dLDYj83e19krxy', cat_list)
```

Validating that this is a picture of your car...

(u'n02930766', u'cab')

'Validation complete - proceed to damage evaluation'

CHAPTER 5

RESULTS AND DISCUSSIONS

This section will provide an overall analysis of what this project has achieved and how this breakthrough can be used in future models as reference, providing a visual representation of the functionality.

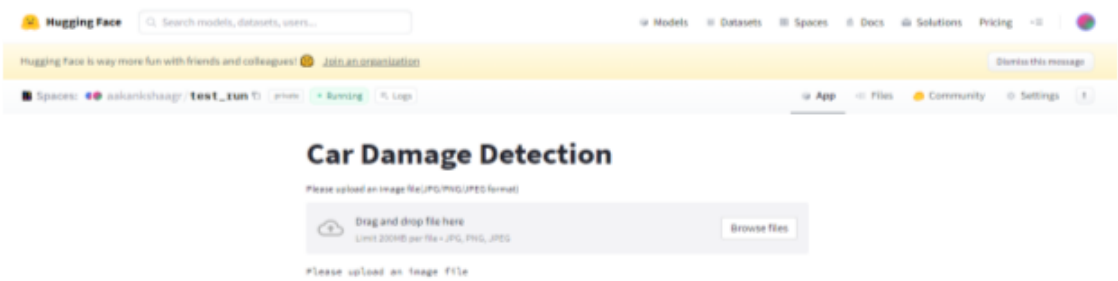


Fig 5.1: Home Page

Fig 5.2: Upload Image

Fig 5.3: Output

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

The work that has been done thus far includes analysis of data about a specific individual or group of individuals in order to predict the likelihood of them filing a claim in the future. Machine learning algorithms can be trained on a large dataset of data points, including the individual's driving history, the type of car they drive, their age and gender, and other factors, in order to make these predictions. The resulting predictions can then be used to calculate personalized insurance premiums for each individual.

AI is also being used to automate the claims process, by analyzing claims data and identifying patterns and trends that can be used to predict the outcome of a claim. This can help to streamline the claims process and improve the efficiency and accuracy of the claimsprocess.

6.2 Future Scope

In the future, it is likely that AI will continue to play a significant role in the car insurance industry, as more and more data becomes available and AI technologies continue to improve. Some potential areas of focus for the development of AI in the car insurance industry include the use of autonomous vehicles, the integration of telematics data, and the development of more advanced machine learning algorithms.

REFERENCES

[1] Deep Learning-Based Car Damage Classification and Detection, Mahavir Dwivedi, HashmatShadab Malik, S. N. Omkar, Edgar Bosco Monis, Bharat Khanna, Satya Ranjan Samal, Ayush Tiwari & Aditya Rathi, Springer, Singapore, 14 August 2020

[2] Image processing based severity and cost prediction of damages in the vehicle body: A computational intelligence approach, W.A. Rukshala Harshani, Kaneeka Vidanage,IEEE, 2017 National Information Technology Conference (NITC)

[3] Deep Learning Based Car Damage Detection, Classification and Severity, Ritik Gandhi, International Journal of Advanced Trends in Computer Science and Engineering October 06,2021

[4] Car Damage Detection and Cost Evaluation Using MASK R-CNN, J. D. Dorathi Jayaseeli, Greeta Kavitha Jayaraj, Mehaa Kanakarajan & D. Malathi, 28 September 2021,Springer, Singapore

[5] Automated vehicle inspection model using a deep learning approach, Mohamed Mostafa Fouad, Karim Malawany, Ahmed Gamil Osman, Hatem Mohamed Amer, Ahmed MohamedAbdulkhalek & Abeer Badr Eldin,Springer, 03 July 20

[6] Damage Assessment of Vehicle Bodies Using a Hybrid Deep Learning Approach" by Haixiang Liu, Xuefeng Chen, Xiaojie Li, Sensors, 2019.

[7] Vehicle Damage Detection and Localization Using Convolutional Neural Networks" by Mahesh P. Nandakumar, Nandakumar Selvaraj, IEEE Transactions on Intelligent Transportation Systems, 2019

[8] Image-Based Vehicle Damage Detection Using a Deep Convolutional Neural Network" by Mingming Wang, Chunhui Zhao, Yanping Cao, Sensors, 2018.