TRAFFIC SIGN OBSERVATION FOR AUTONOMOUS CARS

Dr.M.Kanipriya
Dept.of Computational intelligence
SRM IST
102908
Assistant professor

Nalla Perumal
CSE SWE
SRM IST
RA1911033010012
Chennai, India

Benin Sam
CSE SWE
SRM IST
RA1911033010030
Chennai, India

ABSTRACT - One of most important technology of automated driving systems is to detect traffic signs in various environmental conditions. Due to various environmental changes it is difficult to detect the traffic signs more accurately. In this project, a deep neural network is used for different traffic sign recognition. The presented method uses convolution neural network which extracts more features from input image dataset to carry out recognition. According to statistics, most road accidents take place due to lack of response time to instant traffic events. With self-driving cars, this problem can be addressed by implementing automated systems to detect these traffic events. This involves correctly identifying the traffic signs that can be faced by an automated vehicle, classifying them, and responding to them. The techniques used by us made our system achieve better accuracy under variable lighting conditions. Sequential model can be built by using the Sequential () class. Here, we sequentially add layers to the model using the add()method. According to the Keras documentation, A filter or a kernel in a conv2D layer has a height and a width. They are generally smaller than the input image and so we move them across the whole image processing may also include color.

II. LITERATURE SURVEY

Automatic traffic sign detection and recognition - M Swathi - K. V. Suresh IEEE

Traffic sign detection and classification is one of the main tasks of the advanced driving assistance system (ADAS). It is an integral part of the automatic driving vehicle. How to improve the accuracy and detection speed of traffic sign recognition has always been the focus of research. To solve the above problems, a fast three-stage traffic sign detection and classification method is proposed in this paper to improve the algorithm accuracy. In the first stage, we develop a probability distribution model based on the color, location, and type of traffic signs as a priori information, which can drastically minimize the search range of signs and enhance detection efficiency. In the second stage, this paper proposes an image color segmentation method based on Gaussian mixture model (GMM) as the detection module, uses the YCbCr color model for image segmentation. The morphological closure is then performed to refine the segmented image. In the third stage, the classification module classifies the extracted target areas through deep convolutional neural network (CNN). The drawbacks are Large Dataset and Slow Learn time.

Traffic Sign Recognition

The dataset contains more than 50,000 images of different traffic signs. It is further classified into 43 different classes. The dataset is quite varying, some of the classes have many images while some classes have few images. The size of the dataset is around 300 MB. The dataset has a train folder which contains images inside each class and a test folder which we will use for testing our model. The drawbacks are Large Dataset and medium Learn time.

Traffic Signs Recognition using DenseNet and Keras in Python - Shikha Gupta

The image dataset is consists of more than 50,000 pictures of various traffic signs(speed limit, crossing, traffic signals, etc.) Around 43 different classes are present in the dataset for image classification. The drawbacks are Large Dataset and Slow Learn time.

Automatic vehicle sign recognition system - Zhou Tao - Yang Zaoli

Nonnegative sparse representation has become a popular methodology in medical analysis and diagnosis in recent years. In order to resolve network degradation, higher dimensionality in feature extraction, data redundancy, and other issues faced when medical images parameters are trained using convolutional neural networks. Lung tumors in chest CT image based on nonnegative, sparse, and collaborative representation classification of DenseNet (DenseNet-NSCR) are proposed by this paper: firstly, initialization parameters of pretrained DenseNet model using transfer learning; secondly, training DenseNet using CT images to extract feature vectors for the full connectivity initialization parameters of pretrained DenseNet model using transfer learning; secondly, training DenseNet using CT images to extract feature vectors for the
the method has better robustness and generalization ability through comparison experiment using AlexNet, GoogleNet, and DenseNet-201 models. The drawbacks are Larger data set , High Accuracy and Slow.

Deep Learning Model for Traffic Sign Detection Using Capsule Networks – Dinesh

Convolutional neural networks are the most widely used deep learning algorithms for traffic signal classification till date but they fail to capture pose, view, orientation of the images because of the intrinsic inability of max pooling layer.This paper proposes a novel method for Traffic sign detection using deep learning architecture called capsule networks that achieves outstanding performance on the German traffic sign dataset.Capsule network consists of capsules which are a group of neurons representing the instantiating parameters of an object like the pose and orientation by using the dynamic routing and route by agreement algorithms.unlike the previous approaches of manual feature extraction,multiple deep neural networks with many parameters,our method eliminates the manual effort and provides resistance to the spatial variances.CNNs can be fooled easily using various adversary attacks and capsule networks can overcome such attacks from the intruders and can offer more reliability in traffic sign detection for autonomous vehicles.Capsule network have achieved the state-of-the-art accuracy of 97.6% on German Traffic Sign Recognition Benchmark dataset (GTSRB).

Evaluation of deep neural networks for traffic sign detection systems - Álvaro - Arcos-García

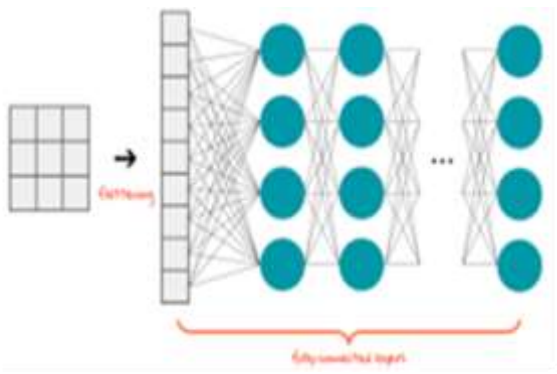
Traffic sign detection systems constitute a key component in trending real-world applications, such as autonomous driving, and driver safety and assistance. This paper analyses the state-of-the-art of several object-detection systems (Faster R-CNN, R-FCN, SSD, and YOLO V2) combined with various feature extractors (Resnet V1 50, Resnet V1 101, Inception V2, Inception Resnet V2, Mobilenet V1, and Darknet-19) previously developed by their corresponding authors. We aim to explore the properties of these object-detection models which are modified and specifically adapted to the traffic sign detection problem domain by means of transfer learning. In particular, various publicly available object-detection models that were pre-trained on the Microsoft COCO dataset are fine-tuned on the German Traffic Sign Detection Benchmark dataset. The evaluation and comparison of these models include key metrics, such as the mean average precision (mAP), memory allocation, running time, number of floating point operations, number of parameters of the model, and the effect of traffic sign image sizes. Our findings show that Faster R-CNN Inception Resnet V2 obtains the best mAP, while R-FCN Resnet 101 strikes the best trade-off between accuracy and execution time. YOLO V2 and SSD Mobilenet merit a special mention, in that the former achieves competitive accuracy results and is the second fastest detector, while the latter, is the fastest and the lightest model in terms of memory consumption, making it an optimal choice for deployment in mobile and embedded devices.

III. MODULES DESCRIPTION AND IMPLEMENTATION

1.Sequential
It helps to group stacks of layers into a model

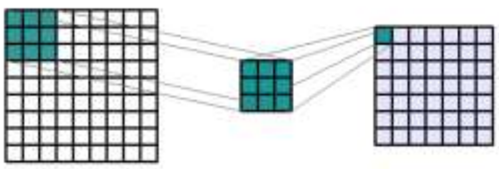
In Keras, a Sequential model can be built by using the Sequential () class. Here, we sequentially add layers to the model using the add()method. According to the Keras documentation,

2.FLATTEN
reduces the input data into a single dimension instead of 2 dimensions



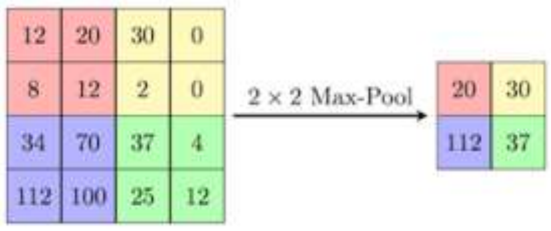
3.Conv2D

A filter or a kernel in a conv2D layer has a height and a width. They are generally smaller than the input image and so we move them across the whole image.



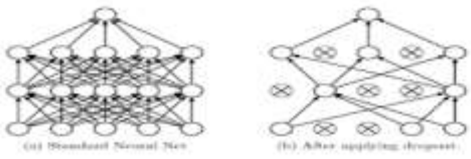
4. Maxpool2D

- Choosing the largest pixel
- Used for compressing the data



5. Dropout

randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting.



IV. SYSTEM ARCHITECTURE AND DESIGN

1.ROI Extraction

It improves the efficiency of the system, Removes undesirable parts of image like background and keeps only the symbol during training to boost accuracy.
In many machine learning applications, image processing is an important step in preparing training data. One of the main tasks in this process is to remove unwanted parts of the image, such as the background, to improve the accuracy of the system. This is done through a

process.Imagepreprocessing consists of many techniques to improve image data quality and remove noise or negative data that can adversely affect the accuracy of machine learning models. One such technique is background subtraction, in which the background of the image is removed to focus on objects or symbols of interest.

Background removal is particularly useful in object detection and recognition tasks where objects or symbols of interest must be accurately identified and isolated. Machine learning models can improve the accuracy of the model by removing the background and focusing only on the object.Another technique used in preview images is image resizing. This includes resizing the image to fit the machine learning model's requirements. Image resizing helps reduce the aspect ratio of the model and improve the quality of the model by reducing the number of pixels in the image.In addition to these techniques, image processing may also include color normalization, contrast, and image filtering, all of which can help improve the image quality and increase accuracy of machine learning models.

In general, image preprocessing is an important step in preparing image data for machine learning. Pre-images can help improve the accuracy and performance of learning models by removing unwanted parts of the image and improving its quality, making them more effective and reliable.

2.Region Processing

Takes care of which region of the image number lettering or a particular feature is located. Regional processing is a technique used in image processing to identify and modify certain regions of an image. This technique is especially useful when working with complex images that contain multiple elements such as text, symbols, and images.

Regional processing involves analyzing image data to identify specific areas of interest. This can be done in a variety of ways, including thresholding, edge analysis, and segmentation. Once an area is defined, it can be modified in various ways, such as cropped, resized, or enhanced.

Area processing is often used in applications such as OCR (Optical Character Recognition) where the goal is to extract text from images. In this case, the area function will be used to identify the specific area of the image where the text is located, and then extract and process the text file. Another application of the

functional area is object recognition, where the goal is to identify specific objects in an image. In this case, the area function is used to identify areas in the image where the object is located and then analyze the data in those areas to determine the type of object. try this.

In general, region processing is an important technique in image processing that allows the identification and processing of specific regions in an image. The machine has many uses and is used in many industries, including medicine, manufacturing and entertainment.

3.Feature Extraction

First layer extract simple features like lines and last layer captures complex features of training sample . Convolutional neural networks (CNNs) are widely used in image recognition and processing and are particularly good at capturing visual patterns and features important for image classification. CNNs typically have multiple layers, each performing a specific task in processing the input image. The first layer of CNN usually removes simple features like lines and edges, while the last layer retains the features of the training samples.

The first layer of a CNN is often called the input layer. The input method takes the raw pixel data of the image and applies a convolutional filter technique to the input data.

Each filter removes certain features from the input image, such as edges, lines, or corners. Convolutional filters are usually small and are applied to the entire input image, one region at a time. The output of the first layer is a set of feature maps representing features extracted from the input image. The second layer of the

CNN is usually a nested layer that reduces the rest of the feature map created by the first layer. This is done by downsampling the feature map using a pooling function such as maximum pooling or average pooling.

CNN's fifth and sixth layers are convolutional processes that often use filters to add maps that capture various aspects of the input image. The first layer of CNN is important for feature extraction because it is responsible for identifying low-level features of the image that can be combined in layers to create more features. The filters in this layer are usually small (eg 3x3 or 5x5) and are applied to the entire input image, one region at a time. Each filter performs a convolution on the input image, multiplies each pixel value by a corresponding weight, and adds the results to create a new pixel value in the output map.

The advantage of using convolutional filters in the first layer is their ability to capture local information such as edges and lines, which are often crucial for image recognition. For example, an edge filter detects the boundaries between light and dark areas of the image, while a line filter detects straight lines at certain angles.

After the first layer creates a series of maps, CNN layers are created on top of these features to capture more patterns and patterns in the input image. Layers after the first layer are used to reduce the spatial resolution of the feature map, making it easier to detect slightly invariant patterns in the input image. For example, maximum pooling takes the maximum value in a small region of a specified map and uses it to represent that region in the next layer.

After pooling layers, convolutional layers use filters, each capturing a different aspect of the input image, in addition to feature maps. These filters are usually larger than those used in the first layer and are designed to capture more diverse patterns. For example, a filter on a background layer can detect a texture or shape that is important for recognizing a particular object.

CNN can stack convolution and pooling layers to create a feature hierarchy that represents the input image at different abstraction levels. The last layer of the network takes these levels and maps them to the class list used to make predictions on the input image.

In summary, the first layer of CNN is responsible for extracting simple features such as lines and edges from the input image. Network layers created by these features to capture more patterns and patterns in the input image. By stacking layers, CNNs can learn hierarchical representation of input images, which is well suited for image recognition and processing. The last layer of the CNN is the output layer, which uses the custom maps created by the layers and uses them to make predictions about the input image. The output method can be used in many different ways, depending on the specific task being performed. For example, in an image classification task, the output process may consist of a group of neurons, each representing a different area. The neuron with the highest activation value represents the predicted class for the input image.

In summary, the first layer of CNN extracts simple features such as lines and edges from the input image, while the last layer captures the features of the training samples. The layers of the network use filters that gradually capture the features of the input image, in addition to the custom maps created by the previous layers. This subtraction is what makes CNNs effective at image recognition and processing.

After the first layer creates a series of maps, CNN layers are created on top of these features to capture more patterns and patterns in the input image. Layers after the first layer are used to reduce the spatial resolution of the feature map, making it easier to detect slightly invariant patterns in the input image. For example, maximum pooling takes the maximum value in a small region of a specified map and uses it to represent that region in the

next layer.

After pooling layers, convolutional layers use filters, each capturing a different aspect of the input image, in addition to feature maps. These filters are usually larger than those used in the first layer and are designed to capture more diverse patterns. For example, a filter on a background layer can detect a texture or shape that is important for recognizing a particular object.

In summary, the first layer of CNN extracts simple features such as lines and edges from the input image, while the last layer captures the features of the training samples. The layers of the network use filters that gradually capture the features of the input image, in addition to the custom maps created by the previous layers. This subtraction is what makes CNNs effective at image recognition and processing which is well suited for image recognition and processing.

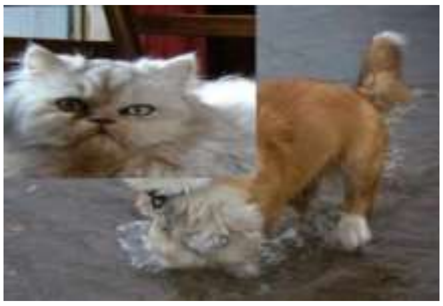
V.CHALLENGES

- Ability to detect the signs in the dark is reduced
- Accuracy can be decreased with bad weather ie (fog , smoke etc)
- Cannot train on different sizes of images
- fail to encode the position and orientation of objects

VI. OBJECTIVES

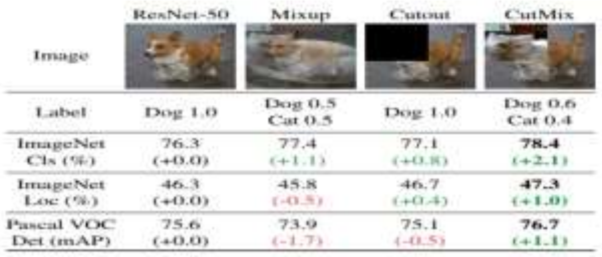
- raining using grey scaled images
- Training using blurred images
- Comparing with different models
- Hybrid approach
- Cutmix

VII. CUTMIX



- Cutmix is a new technology of data augmentation
- Before cutmix regional drop out strategies where used
- In regional drop out random parts of images is replaced by a patch as shown below .

Cutmix



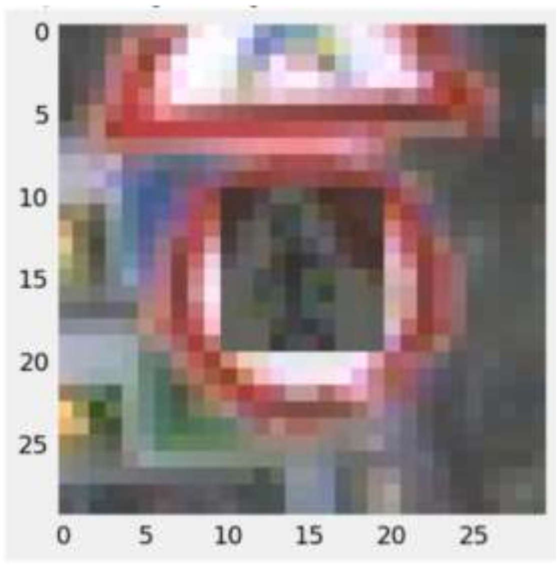
	ResNet-50	Mixup	Cutout	CutMix
Image				
Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4
ImageNet Cls (%)	76.3 (+0.0)	77.4 (+1.1)	77.1 (+0.8)	**78.4** (+2.1)
ImageNet Loc (%)	46.3 (+0.0)	45.8 (-0.5)	46.7 (+0.4)	**47.3** (+1.0)
Pascal VOC Det (mAP)	75.6 (+0.0)	73.9 (-1.7)	75.1 (-0.5)	**76.7** (+1.1)

- In CutMix augmentation we cut and paste random patches between the training images.
- The ground truth labels are mixed in proportion to the area of patches in the images.
- No information loss compared to regional drop outs .

- CutMix increases localization ability by making the model to focus on less discriminative parts of the object being classified and hence is also well suited for tasks like object detection.

Cutmix Algorithm

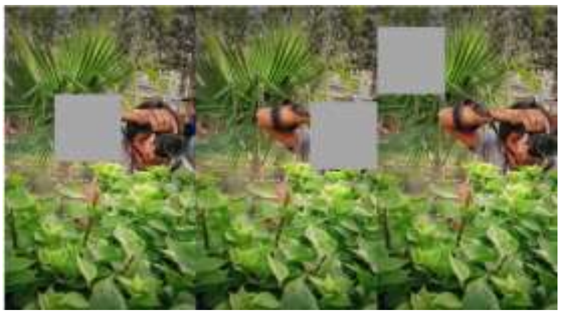
- Let x be an image of shape W*H*C where W, H and C are Width, Height and number of channels respectively and y be the ground truth label. We combine two samples (x_a, y_a) and (x_b, y_b) to produce a new sample (x_c, y_c). The equation can be given as
- M is the binary mask (shows where the image is replaced)
- λ value can be taken between 0 and 1



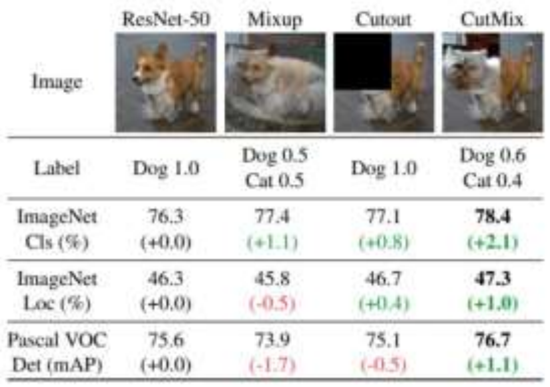
Cutmix : With the help of cutmix the models accuracy can be increased without adding noise like used to be done while following regional drop outs which might result in loss of certain features while training . In cutmix the idea is instead of introducing a noise while training another training image is cut and the labels are mixed Example shown below

Cut mix is a new technology in data augmentation

Example of regional drop out :



Here model can concentrate on less discriminative parts of an object but I was found out that it could result in information loss as a result cutmix was introduced.



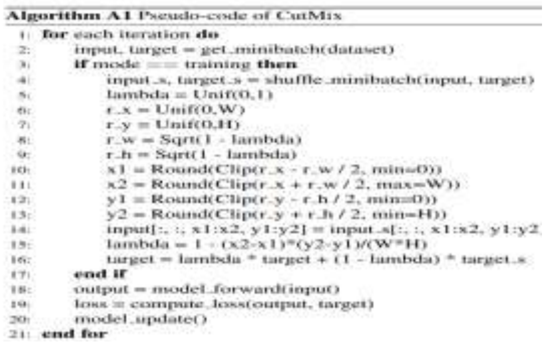
	ResNet-50	Mixup	Cutout	CutMix
Image				
Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4
ImageNet Cls (%)	76.3 (+0.0)	77.4 (+1.1)	77.1 (+0.8)	**78.4** **(+2.1)**
ImageNet Loc (%)	46.3 (+0.0)	45.8 (-0.5)	46.7 (+0.4)	**47.3** **(+1.0)**
Pascal VOC Det (mAP)	75.6 (+0.0)	73.9 (-1.7)	75.1 (-0.5)	**76.7** **(+1.1)**

Cut mix has been noticed to have higher accuracies compared to other similar strategies.

VIII. CUTMIX ALL STEPS INVOLVED

In CutMix augmentation we cut and paste random patches between the training images.
The ground truth labels are mixed in proportion to the area of patches in the images.
No information loss compared to regional drop outs .
CutMix increases localization ability by making the model to focus on less discriminative parts of the object being classified and hence is also well suited for tasks like object detection.

Cutmix algorithm



Algorithm A1 Pseudo-code of CutMix

```
1:  for each iteration do
2:      input, target = get_minibatch(dataset)
3:      if mode == training then
4:          input_s, target_s = shuffle_minibatch(input, target)
5:          lambda = Unif(0,1)
6:          r_x = Unif(0,W)
7:          r_y = Unif(0,H)
8:          r_w = Sqrt(1 - lambda)
9:          r_h = Sqrt(1 - lambda)
10:         x1 = Round(Clip(r_x - r_w / 2, min=0))
11:         x2 = Round(Clip(r_x + r_w / 2, max=W))
12:         y1 = Round(Clip(r_y - r_h / 2, min=0))
13:         y2 = Round(Clip(r_y + r_h / 2, min=H))
14:         input[:, :, x1:x2, y1:y2] = input_s[:, :, x1:x2, y1:y2
15:         lambda = 1 - (x2-x1)*(y2-y1)/(W*H)
16:         target = lambda * target + (1 - lambda) * target_s
17:     end if
18:     output = model_forward(input)
19:     loss = compute_loss(output, target)
20:     model_update()
21: end for
```

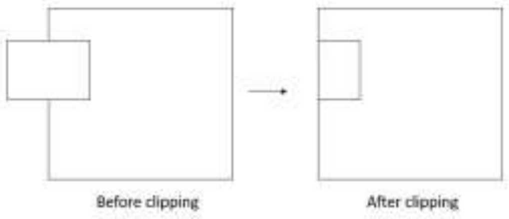
If the offset is set in such a way that a part of the cutout exceeds the image clipping is performed. The code to perform the clipping function is given below.

```
boundaryx1 = tf.clip_by_value(cut_x[0] -
 cut_w // 2, 0, IMG_SIZE)
   boundaryy1 = tf.clip_by_value(cut_y[0]
 - cut_h // 2, 0, IMG_SIZE)
   bbx2 = tf.clip_by_value(cut_x[0] + cut_
w // 2, 0, IMG_SIZE)
   bby2 = tf.clip_by_value(cut_y[0] + cut_
h // 2, 0, IMG_SIZE)
  target_h = bby2 - boundaryy1
```

where // is the floor division
example 15//2 = 7

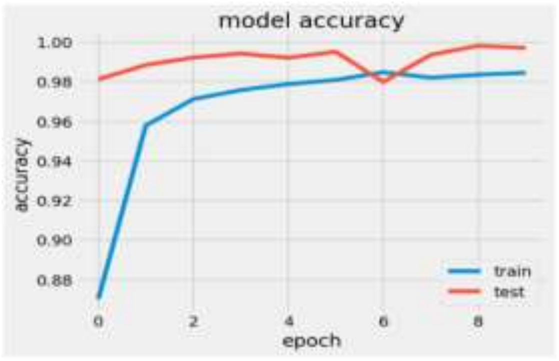
while in normal division
15/2 = 7.5

Here this is done to prevent typecast error while working with the tensor .



IX. RESULTS AND DISCUSSIONS

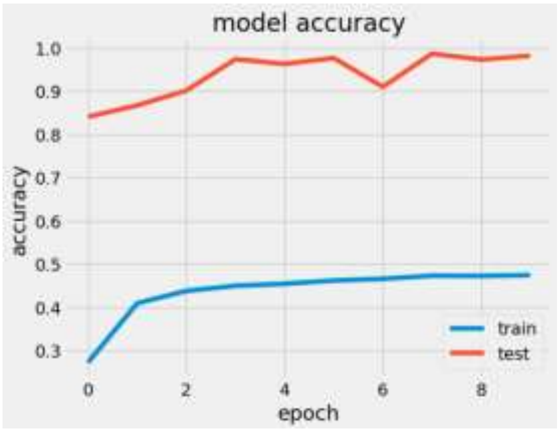
With the help of cutmix in our project we where able to get higher accuracy in our project



Above image is the test and train stats on each epoch for an ai which used cutmix method.

CutMix enables the network to learn more powerfully and broadly. It exposes the network to various training examples, forcing them to combine information from different images. This can help the network become more suitable for new images, go unnoticed and reduce overfitting.

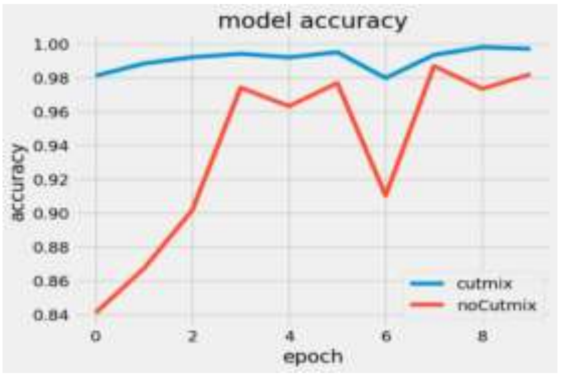
Various studies have shown that using CutMix during training improves the accuracy of various computer tasks, including image classification, object detection, and semantic segmentation. CutMix is particularly good in situations with limited data, as it allows the network to learn from many models and achieve better generalization.



Above image is the test and train stats on each epoch for an ai which doesn’t use cutmix method.

If CutMix is not used during training, the network will not be able to learn well from the available data, which will reduce accuracy. This is especially true where the dataset is limited or the images are similar to each other because the network may not have been exposed to enough training data to learn features that generalize well.

Therefore, it can be concluded that "without CutMix the accuracy is less" means that the CNN model trained without CutMix will have less accuracy than the same model trained with CutMix.



Above image is the cutmix validation accuracy and no cutmix validation accuracy stats on each epoch for an ai which used cutmix method.

X.REFERENCES

[1] M. Swathi and K. V. Suresh, "Automatic traffic sign detection and recognition: A review," *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)*, Chennai, India, 2017, pp. 1-6, doi: 10.1109/ICAMMAET.2017.8186650. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8186650&isnumber=8186622

[2] S.Gupta, "Traffic Signs Recognition using CNN and Keras in Python," Analytics Vidhya, Dec. 22, 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/12/traffic-signs-recognition-using-cnn-and-keras-in-python/. [Accessed feb 3, 2023].

[3] S.Nath "CutMix for Improved Training of Deep Neural Networks for Image Classification," Keras Documentation, [Online]. Available: [https://keras.io/examples/vision/cutmix/.](https://keras.io/examples/vision/cutmix/) [Accessed feb 3, 2023].

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A.L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 4, pp. 834-848, Apr. 2018. https://ieeexplore.ieee.org/document/8100164 [Accessed feb 7, 2023].

[5] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016. https://ieeexplore.ieee.org/document/7780671 [Accessed March 14, 2023].

[6] R. Takahashi, T. Matsubara, and K. Uehara, "RICAP: Random Image Cropping and Patching Data Augmentation for Deep CNNs," in Asian Conference on Machine Learning, 2018, https://www.sciencedirect.com/science/article/pii/S2590005622000911 [Accessed April 17,2023].

[7] D. Dwibedi, T. Misra, and S. Hebert, "Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, https://ieeexplore.ieee.org/document/8099963 [Accessed April 20,2023].