# **Malnad College of Engineering**

(An Autonomous Institution under Visvesvaraya Technological University, Belagavi)

Hassan - 573 202



**Subject:Deep Learning Subject Code:18IS864** 

# **ACTIVITY 01**

## VEHICLE CLASSIFICATION USING GOOGLENET

Submitted by
Harshavardhan T P (4MC18IS011)
Sanjith B V (4MC18IS041)
Abdul Azeem (4MC19IS400)

Submitted to Dr.Balaji Prabhu Associate Professor

Department of Information Science & Engineering Malnad College of Engineering Hassan - 573 202

2021-22

#### INTRODUCTION

It is easier to use pre-trained convolutional neural network to classify images than to build one from scratch. The best thing about pre-trained CNN is they are well optimized. For example — GoogleNet. It is one of the most popular convolutional neural networks. It has been trained to recognize 1000 different objects. What if you need to train it to classify your own data? Well, this is exactly what has been covered in this article. This article shows the process of vehicle classification using GoogleNet convolutional neural network (CNN).

GoogLeNet is a convolutional neural network that is 22 layers deep. You can load a pretrained version of the network trained on either the ImageNet or Places365 data sets. The network trained on ImageNet classifies images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. The network trained on Places365 is similar to the network trained on ImageNet, but classifies images into 365 different place categories, such as field, park, runway, and lobby. These networks have learned different feature representations for a wide range of images.

GoogLeNet is a 22-layer deep convolutional neural network that's a variant of the Inception Network, a Deep Convolutional Neural Network developed by researchers at Google. The GoogLeNet architecture presented in the ImageNet Large-Scale Visual Recognition Challenge 2014(ILSVRC14) solved computer vision tasks such as image classification and object detection find out how well it performed at the conclusion section of this article.

Today GoogLeNet is used for other computer vision tasks such as face detection and recognition, adversarial training etc.

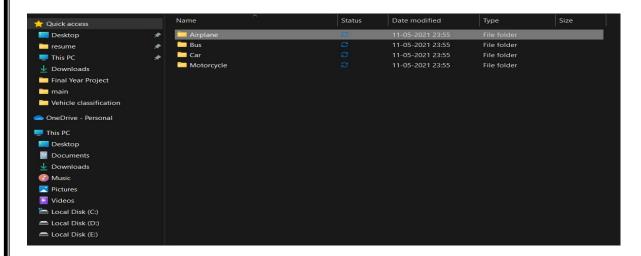
### **Procedure**

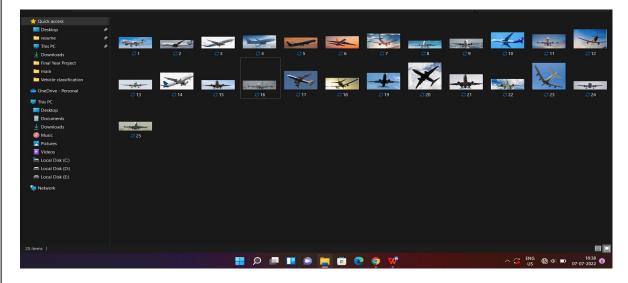
The procedure is pretty simple. Actually, the origin of idea of transfer learning has been originated from the need of simplicity and faster deployment. It takes long to design a convolutional neural network and even longer to optimize it. On the other hand, using transfer learning, you can retrain an existing optimized convolutional neural network almost instantly. Here are the five steps to apply transfer learning to classify vehicle using GoogleNet

#### Load your vehicle dataset,

- 1. Split the dataset into training, validation and testing dataset,
- 2. Resize the images according to the input layer of the network,
- 3. Modify the feature learner and output layer of the GoogleNet,
- 4. Re-train the feature learner layer using your dataset.

#### **Dataset**





We have made 4 classes of data which are airplane, bus, car, motorcycle

Contains the images of there respective class which are downloaded by google.

#### Code

The concept of vehicle classification using convolutional neural network (CNN) has been implemented using two separate scripts. They are:

- 1. Training To train the network
- **2.** Test Network To test the network

# **Code of Training Script:**

```
Dataset = imageDatastore('Dataset', 'IncludeSubfolders',
true, 'LabelSource', 'foldernames');
[Training Dataset, Validation Dataset, Testing Dataset] =
splitEachLabel(Dataset, 0.7, 0.15, 0.15);
net = googlenet;
analyzeNetwork(net)
Input Layer Size = net.Layers(1).InputSize(1:2);
Resized Training Dataset =
augmentedImageDatastore(Input Layer Size,Training Dataset);
Resized Validation Dataset =
augmentedImageDatastore(Input Layer Size, Validation Dataset);
Resized Testing Dataset =
augmentedImageDatastore(Input Layer Size,Testing Dataset);
Feature Learner = net.Layers(142).Name;
Output Classifier = net.Layers(144).Name;
Number of Classes = numel(categories(Training Dataset.Labels));
New Feature Learner = fullyConnectedLayer(Number of Classes, ...
       'Name', 'Vehicle Feature Learner', ...
        'WeightLearnRateFactor', 10,...
        'BiasLearnRateFactor', 10);
New Classifier Layer = classificationLayer('Name', 'Vehicle Classifier');
Network Architecture = layerGraph(net);
```

```
New Network = replaceLayer(Network Architecture, Feature Learner,
New Feature Learner);
New Network = replaceLayer(New Network, Output Classifier,
New Classifier Layer);
analyzeNetwork(New Network)
Minibatch Size = 4;
Validation Frequency =
floor(numel(Resized Training Dataset.Files)/Minibatch Size);
Training Options = trainingOptions('sgdm', ...
      'MiniBatchSize', Minibatch Size, ...
      'MaxEpochs', 6, ...
      'InitialLearnRate', 3e-4, ...
      'Shuffle', 'every-epoch', ...
      'ValidationData', Resized Validation Dataset, ...
      'ValidationFrequency', Validation Frequency, ...
      'Verbose', false, ...
      'Plots', 'training-progress');
net = trainNetwork(Resized Training Dataset, New Network,
Training Options);
Code of Test Network Script:
[Predicted Label, Probability] = classify(net, Resized Testing Dataset);
accuracy = mean(Predicted Label == Testing Dataset.Labels);
index = randperm(numel(Resized Testing Dataset.Files), 4);
figure
for i = 1:4
   subplot(2,2,i)
   I = readimage(Testing Dataset, index(i));
   imshow(I)
   label = Predicted Label(index(i));
   title(string(label) + ", " + num2str(100*max(Probability(index(i), :)), 3) +
"%");
end
```

#### **Screenshots: Analysis Network** Deep Learning Network Analyzer Analysis for trainNetwork usage Name: New\_Network 5.9M 144 0 🛕 0 0 Analysis date: 07-Jul-2022 18:57:29 ANALYSIS RESULT 0 Name Activations 224(S) × 224(S) × 3(C) × 1(B) Image Input conv1-7x7\_s2 112(S) × 112(S) × 64(C) × 1(B) conv1-relu\_7x7 Weig... $7 \times 7 \times 3$ .. Bias $1 \times 1 \times 64$ pool1-3x3\_s2 conv1-relu\_7x7 ReLU 112(S) × 112(S) × 64(C) × 1(B) pool1-norm1 pool1-3x3\_s2 3×3 max pooling with stride [2 2] and pa. 56(S) × 56(S) × 64(C) × 1(B) conv2-3x3\_r.. pool1-norm1 cross channel normalization with 5 chan... Cross Channel Nor 56(S) × 56(S) × 64(C) × 1(B) 6 conv2-3x3\_reduce 64 1×1×64 convolutions with stride [1 1] ... 56(S) x 56(S) x 64(C) x 1(B) conv2-3x3 Weig... $1 \times 1 \times 64$ ... Bias $1 \times 1 \times 64$ conv2-relu\_3x3 conv2-relu\_3x3\_reduce ReLU 56(S) × 56(S) × 64(C) × 1(B) conv2-norm2 conv2-3x3 192 3×3×64 convolutions with stride [1 1... 56(S) × 56(S) × 192(C) × 1(B) Convolution Weig... 3 x 3 x 64 . Bias 1 x 1 x 192 pool2-3x3\_s2 9 conv2-relu\_3x3 56(S) × 56(S) × 192(C) × 1(B) inception\_3a-... inception\_3a-... inception\_3ainception\_3a-. • inception\_3a-. • inception\_3a-Cross Channel Nor... 56(5) × 56(5) × 192(C) × 1(B) 10 conv2-norm2 inception\_3a-... inception\_3a-... inception\_3a-. pool2-3x3\_s2 3×3 max pooling with stride [2.2] and pa... 28(S) × 28(S) × 192(C) × 1(B) inception\_3ainception\_3a-1x1 64 1×1×192 convolutions with stride [1 1... Convolution 28(5) × 28(5) × 64(C) × 1(B) Weig... $1 \times 1 \times 192$ ... Bias $1 \times 1 \times 64$ • Inception\_3ainception\_3a-relu\_1x1 ReLU 28(S) × 28(S) × 64(C) × 1(B) inception\_3b-... inception\_3b-... inception\_3binception\_3a-3x3\_reduce 96 1×1×192 convolutions with stride [1 1... Convolution 28(S) × 28(S) × 96(C) × 1(B) Weig... 1 × 1 × 192... Bias 1 × 1 × 96 inception\_3b-... inception\_3b-... inception\_3b-... ReLU 28(S) x 28(S) x 96(C) x 1(B) inception\_3a-relu\_3x3\_reduce inception\_3b-... inception\_3b-... inception\_3a-3x3 128 3×3×96 convolutions with stride [1 1. 28(S) × 28(S) × 128(C) × 1(B) Weig... $3 \times 3 \times 96$ ... Bias $1 \times 1 \times 128$ inception\_3binception\_3a-relu\_3x3 ReLU 28(5) × 28(5) × 128(C) × 1(B) pool3-3x3 s2 🧨 🖫 🥠 ^ \$\rightarrow\$ ENG \$\pi\$ \$\phi\$ \$\p Q 🔳 🗓 0 **Training Progress** Training Progress (07-Jul-2022 18:58:52) Results Training Progress (07-Jul-2022 18:58:52) Validation accuracy: 100.00% Max epochs completed Training finished: Training Time 07-Jul-2022 18:58:52 Elapsed time: 3 min 23 sec Training Cycle Epoch: 6 of 6 Iteration: 102 of 102 60 Iterations per epoch: Maximum iterations: 102 50 Validation 40 17 iterations Frequency Other Information Single CPU Hardware resource: 0.0003 Export Training Plot Learn more Iteration \_\_\_\_ Training - - - Validation ----- Training (smoothed) ----- Training ^ G ENG ( □ 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 19:02 ( 1 Q 🔳 🗓

