Report

Preprocessing the Data

1) The data was obtained from https://docs.google.com/forms/d/e/1FAIpQLSfuMMGafmiZ35alIgYkZeyGkR6gHhBURjxJPSe6aB6CWjN1EA/viewform under the under the Open Data Commons Attribution License: https://opendatacommons.org/licenses/by/1.0/

2) The dataset was split into two parts training and validation data. The validation data was then split into two equal parts for the purpose of validation and testing. The dataset contains two classes, damaged and whole.

3) Images have been preprocessed using ImageDataGenerator in Keras. The value of each pixel is normalized between 0 and 1 to be able to feed it into the model.

Training the Model

For fast computation and low training time a modified version of VGG16 was chosen as the classification model.

The VGG16 model is modified as follows –

1) The original VGG16 model is first loaded with its weights pertained on the imagenet dataset.

2) The top classifier of the model is removed.

3) The penultimate layer of the model is flattened.

4) A new Global Average Pooling 2D Layer is inserted after the flattened layer. As opposed to using a dense layer, pooling layer was used in order to reduce the number of new parameters that the model has to learn in order to make a prediction.

5) A dense layer of size 1 is inserted with sigmoid activation function for performing the classification.

We used transfer learning for performing classification as opposed to training a model from scratch. This was done majorly due to two reasons.

1) As our dataset was small, a model trained from scratch would not be able to learn the deeper contextual features needed for classifying a car as damaged or not.
2) As the images present in the imagenet dataset and the cars dataset our very similar, the weights present in the pretrained can be effective in understanding the contextual features of the images in the cars dataset.

Experimental Results

For the purpose of training the model Google Colab was used.

The model was trained only for 10 epochs because of shortage of gpu and time.

After training for 10 epochs the model was tested on the testing dataset. We achieved **88.57% percent test accuracy.**

The graphs for training accuracy, validation accuracy, training loss and validation loss are present in the notebook.

Explanations

We tried generating explanations as to how the model was classifying an image. We used two different packages.

1) For generating the explanations we used the deep explain package https://github.com/marcoancona/DeepExplain. The package allows us to use different methods such as Saliency, Gradient * Input, Integrated Gradients, epsilon-LRP, DeepLIFT and occlusion. Though the package supports both keras and tensorflow, there are currently a few problems in its keras application when multichannel images as used as it is currently under development. The various visuals obtained have been attached in the

notebook. Though the results were not great, we could distinguish a little as to which parts of the car the model is trying to focus on.

2) We used another package called keras-vis for generating saliency maps. The results with it were much better and distinguishable. The results obtained have been attached in the notebook. The visuals show that the model is trying to focus on the damaged parts of the model.