Deep Learning for Vehicle Recognition

DATA MINING (COSC2111)

Charles Galea (s3688570)
RMIT | MASTER OF DATA SCIENCE

Table of Contents

Deep Learning for Vehicle Recognition

Abstract

We have generated an image recognition convolution neural network that performs well for identifying cars in a range of images. The network was trained against a dataset containing images of cars or cars involved in accidents. Additional random images were included to increase the generality of the classifier. Transfer learning utilizing the VGG16 CNN classifier did not significantly improve the performance of the network. Visual examination of feature maps pertaining to each filter for individual layers of the network indicated that the model generated in these studies retained significant information from the input images.

Introduction

CNN classifier

Convolutional layers

In a convolution neural network convolution layers act as image filters that extract feature (also called activation) maps that highlight the location and strength of detected features in an input image [1]. The stacking of convolutional layers allows for a hierarchical decomposition of the input image where the initial convolutional layers extract features that consist of combinations of lower-level features, such as features comprising multiple lines that form a shape while deeper layers extract higher-level features, such as faces, houses, cars, etc.

Dropout

Dropout is a regularization technique where randomly selected nodes are ignored during training [1]. Subsequently, their contribution to the activation of downstream nodes is temporally removed. During network learning some node weights tuned to specific features can ‘memorize’ images (often referred to as complex adaptations) leading to a fragile model too specialized to the training data. Randomly dropping out nodes during training avoids this problem by forcing other nodes to step in to compensate for the missing nodes. The network becomes less sensitive to specific node weights and results in a network that is more generalized and less prone to overfit the training data.

Data augmentation

Data augmentation is a method to generate more training data from an existing image dataset. Images in the training dataset are processed using a number of random transformations (e.g. rotation, shear, zoom, flip, etc). This produces a larger dataset, which ensures that the network does not see the same image twice and helps to build a more generalized model.

Transfer learning

Transfer learning is a highly efficient and effective method of leveraging a pre-trained network for classifying a relatively small image datasets [1, 2]. The pre-trained network has been previously trained on a large dataset such that the spatial feature hierarchy learned by the network can effectively act as a generic model for our smaller dataset. In this assignment we have used the VGG16 CNN classifier [3] that has previously been trained on the ImageNet dataset which contains 1.4 million labelled images and 1000 different classes.

Two methods can be used to make use of a pre-trained network: feature extract and fine-tuning [1]. Feature extraction consists of using the representations learned by a network to extract interesting features from a new sample. The features are used to train a new classifier. The level of generality of the representations extracted by specific convolutional layers depends on the depth of the layer in the model. Earlier layers extract local, highly generic feature maps (e.g. edges, colours and textures) while layers higher-up extract more abstract concepts (e.g. ear of elephant). Fine-tuning consists of unfreezing a few of the top layers of a frozen model base used for feature extraction and training both the newly added part of the model and these top layers.

Visualizing intermediate activations

Intermediate activations are useful for defining how successive convolutional network layers filter and transform the input images and determining the effect of individual convolutional filters [1]. Visualizing intermediate activation maps consists of displaying the feature maps that are output by various convolutional and pooling layers in the network. This provides an indication of how the input is decomposed into different features.

Problem Statement

The goal of this assignment was to develop a deep learning CNN-based image recognition model to classify images as containing damaged or intact cars.

Dataset

The cars dataset obtained from Prince Hemrajani (RMIT) and containing 3896 images of intact (1948 images) or damaged (1948 images) cars was used to train and validate a convolutional neural network (CNN) classifier.

Methodology

Data pre-processing

The dataset was filtered using a python script to remove corrupted images. The images were then processed to rescale the pixel values to between 0 and 1 and re-sized to a uniform number of pixels prior to feeding into the CNN classifier. All pre-processing steps were undertaken in the programming language Python (Python 3.6; Python Software Foundation, Wilmington, Del.;2009) using the package numpy (https://numpy.org/).

Data augmentation using the Keras ImageDataGenerator function was used to generate more training data by subjecting images in the existing dataset to a number of defined transformations.

Model training

The CNN-based classifier was trained and tested (validated) using 75% and 20% of the pre-processed dataset, respectively. The remaining 5% of the dataset was used to test the trained classifier.

Testing and debugging of the CNN classifier was performed on a local machine with a Windows 10 operating system, a six-core i7 2.7-GHz processor (Intel, Santa Clara, Calif) and 8 GB of RAM using a small subset of the entire dataset. Training and classification of the entire data was performed in either Google Colab using a dual Intel Xeon 2.2-GHz processor and a NVIDIA Tesla K80 graphical processing unit (Nvidia Corporation, Santa Clara, Calif) or on the Google Cloud Deep Learning platform using a NVIDIA Tesla P100 graphical processing unit. The packages Python 3.6.3 (https://www.python.org/), Tensorflow 1.10.0 (https://www.tensorflow.org/), Keras 2.2.2 (https://pypi.org/project/Keras/), CUDA 9.2.148.1 (https://developer.nvidia.com/cuda-zone) and cuDNN 7.2.1 (https://developer.nvidia.com/cudnn) were utilized for these deep learning studies.

The input shape was 128 x 128 x 3 with a binary output and the input images were standardized before being used for training. The CNN network consisted of a single input convolution layer, multiple hidden layers, a dense fully connected layer and an output layer. Adam (an adaptive learning rate optimization algorithm) was used for training the convolutional neural network [4].

Transfer learning using a pre-trained classifier

Feature Extraction

Weights and model architecture for the VGG16 CNN classifier previously trained on the ImageNet dataset were used to extract features for the car dataset. Part of the VGG16 CNN network excluding the final dense fully connected layers (the convolutional base) was used to train a smaller dataset. The VGG16 convolutional base was frozen and a new dense fully connected layer was added. The images are trained using the weights for the previously trained VGG16 convolutional base and the new dense layer [1].

Fine Tuning

Fine tuning is similar to feature extraction method except one or more of the layers of the frozen VGG16 convolutional base are unfrozen and the images are also trained on these layers [1].

Model interpretation and data visualization

Feature (or 'activation') maps for each convolutional and pooling layer in the network were extracted to gain insight into how the network derived its decisions. The feature map indicates how the input is decomposed by different filters learned by the network.

Results (please see appendices for code)

CNN model optimization

The pre-processed images were passed through a stack of convolutional layers containing filters with a small receptive field: 3 x 3 (the smallest size required to capture the various regions of the image – left/right, up/down and centre). Spatial pooling was carried out by max-pooling layers, performed over a 2 x 2 pixel window, following the convolutional layers.

The initial CNN configuration consisted of a convolutional input layer and a convolutional hidden layer followed by a dense (fully connected) layer with 128 channels and a soft-max output layer (configuration A in Table 1). The input and hidden layers are equipped with the rectification (ReLU) non-linearity [5]. The training data was processed in batches containing 32 64 x 64 pixel RGB images with binary labels.

The plots of model loss and accuracy (Fig. 1A) for the initial CNN network (config. A in Table 1) indicated that the model was overfitting. The model performed significantly better on the training data than the validation dataset (Table 3). The training accuracy increased linearly to reach a maximum of 100% within the first 10 epochs while the validation accuracy reached a maximum accuracy of about 75-80% within the initial few epochs and stalls. The validation loss reaches its minimum within the first 5 epochs while the training loss keeps decreasing to a minimum close to 0 within 10 epochs. The observed overfitting is likely due to the relatively low number of training samples (2890) in the dataset.

We initially determined the effect of increasing the image resolution. The size of the images (64 x64) used in the initial model was significantly less than the original image size (512 x 512) possibly resulting in a substantial loss of information. To examine this, we re-ran the model using images with a size of 128 x 128 pixels (configuration B in Table 1). The resulting loss and accuracy plots (Fig. 1B) again exhibited pronounced overfitting indicating that image size had no significant effect.

Several methods such as dropout, weight decay (L2 regularization) and data augmentation can be used to mitigate overfitting. Adding dropout regularization to each convolution layer and to the Dense layer, with a dropout probability of 25% (configurations C, D and F in Table 1), did not improve the model performance and overfitting was still significant in each case (Figs. 1C, 1D and 1F).

We explored the effect of inserting of an additional hidden convolution layer with dropout to determine what effect this would have on overfitting. However, this larger network (configuration E in Table 1) did not lead to reduced overfitting (Fig. 1E).

To determine whether the relatively small size of the dataset was contributing to the overfitting due to 'memorization of the data' we added data augmentation (shear, zoom and horizontal flip) to the model (configuration G in Table 1). This resulted in an improvement in model performance. Compared to the previous model, loss dropped from about 1.39 to 0.60 while accuracy increased from 0.76 to 0.81 (Fig. G and Table 3).

Changing of the number of validation steps from 50 to 32 (configuration H in Table 2) lead to negligible change in model loss or accuracy but decreased the time for training.

The addition of an another hidden convolutional layer with dropout (25% probability) (configuration L in Table 2) resulted in a gain in validation accuracy from 0.82 to 0.86 and decrease in loss from 0.48 to 0.33 (Table 3).

We also considered whether shuffling the dataset prior to training would improve the model (configuration M in Table 2). This resulted in a significant improvement with no apparent overfitting in the model (Fig. 1M). The training and validation curves were closely tracking each other while the final validation loss and gain (0.33 and 0.86, respectively) were similar to the training values (Table 3).

Finally, incorporation of a L1 regularization into the second convolutional layer lead to a further improvement in the performance of the model (configuration N in Table 2).

Table 1. CNN configurations. The depth of the configuration increases from left to right (A to G) as more layers are added (added layer are coloured in red). Conv2D – convolutional layer, Maxpool2D – pooling layer.

	CNN Network Configuration						
Parameters	**A**	**B**	**C**	**D**	**E**	**F**	**G**
Image Size	64 x 64	**128 x 128**	128 x 128	128 x 128	128 x 128	128 x 128	128 x 128
Epochs	50	50	50	50	50	50	50
Val steps	156	156	156	156	156	156	156
Conv layer	Conv2D	Conv2D	Conv2D	Conv2D	Conv2D	Conv2D	Conv2D
Input_shape	(64, 64, 3)	(128, 128, 3)	(128, 128, 3)	(128, 128, 3)	(128, 128, 3)	(128, 128, 3)	(128, 128, 3)
Activation	relu	relu	relu	relu	relu	relu	relu
Pooling	Maxpool2D	Maxpool2D	Maxpool2D	Maxpool2D	Maxpool2D	Maxpool2D	Maxpool2D
Dropout			**Dropout (0.25)**	Dropout (0.25)	Dropout (0.25)	Dropout (0.25)	Dropout (0.25)
Conv layer					**Conv2D**	Conv2D	Conv2D
Activation					**relu**	relu	relu
Pooling					**Maxpool2D**	Maxpool2D	Maxpool2D
Dropout					**Dropout (0.25)**	Dropout (0.25)	Dropout (0.25)
Conv layer							
Activation							
Pooling							
Dropout							
Conv layer	Conv2D	Conv2D	Conv2D	Conv2D	Conv2D	Conv2D	Conv2D
Activation	relu	relu	relu	relu	relu	relu	relu
Pooling	Maxpool2D	Maxpool2D	Maxpool2D	Maxpool2D	Maxpool2D	Maxpool2D	Maxpool2D
Dropout				**Dropout (0.25)**	Dropout (0.25)	Dropout (0.25)	Dropout (0.25)
Flattening	Flattening	Flattening	Flattening	Flattening	Flattening	Flattening	Flattening
Dense layer	Dense(128)	Dense(128)	Dense(128)	Dense(128)	Dense(128)	Dense(128)	Dense(128)
Activation	relu	relu	relu	relu	relu	relu	relu
						Dropout (0.25)	Dropout (0.25)
Output layer	Dense (output = 1)	Dense (output = 1)	Dense (output = 1)	Dense (output = 1)	Dense (output = 1)	Dense (output = 1)	Dense (output = 1)
Activation	softmax	softmax	softmax	softmax	softmax	softmax	softmax
Image aug	No	No	No	No	No	No	**Yes**

Table 2. CNN configurations. The depth of the configuration increases from left to right (H to N) as more layers are added (added layer are coloured in red). Conv2D – convolutional layer, Maxpool2D – pooling layer.

	CNN Network Configuration						
Parameters	**H**	**I**	**J**	**K**	**L**	**M**	**N**
Image Size	128 x 128	128 x 128	128 x 128	128 x 128	128 x 128	128 x 128	128 x 128
						shuffle	shuffle
Epochs	50	50	50	50	50	50	50
Val steps	**32**	32	32	32	32	32	32
Conv layer	Conv2D	Conv2D	Conv2D	Conv2D	Conv2D	Conv2D	Conv2D
Input_shape	(128, 128, 3)	(128, 128, 3)	(128, 128, 3)	(128, 128, 3)	(128, 128, 3)	(128, 128, 3)	(128, 128, 3)
Activation	relu	relu	relu	relu	relu	relu	relu
Pooling	Maxpool2D	Maxpool2D	Maxpool2D	Maxpool2D	Maxpool2D	Maxpool2D	Maxpool2D
Dropout	Dropout (0.25)	Dropout (0.25)	Dropout (0.25)	Dropout (0.25)	Dropout (0.25)	Dropout (0.25)	Dropout (0.25)
Conv layer	Conv2D	Conv2D	**Conv2D**	Conv2D	Conv2D	Conv2D	Conv2D
Activation	relu	relu	**relu**	relu	relu	relu	relu
							L1 reg (0.0005)
Pooling	Maxpool2D	Maxpool2D	**Maxpool2D**	Maxpool2D	Maxpool2D	Maxpool2D	Maxpool2D
Dropout	Dropout (0.25)	Dropout (0.25)		**Dropout (0.25)**	Dropout (0.25)	Dropout (0.25)	Dropout (0.25)
Conv layer					**Conv2D**	Conv2D	Conv2D
Activation					**relu**	relu	relu
Pooling					**Maxpool2D**	Maxpool2D	Maxpool2D
Dropout					**Dropout (0.25)**	Dropout (0.25)	Dropout (0.25)
Conv layer	Conv2D		Conv2D	Conv2D	Conv2D	Conv2D	Conv2D
Activation	relu		relu	relu	relu	relu	relu
Pooling	Maxpool2D		Maxpool2D	Maxpool2D	Maxpool2D	Maxpool2D	Maxpool2D
Dropout	Dropout (0.25)		Dropout (0.25)	Dropout (0.25)	Dropout (0.25)	Dropout (0.25)	Dropout (0.25)
Flattening	Flattening	Flattening	Flattening	Flattening	Flattening	Flattening	Flattening
Dense layer	Dense(128)	Dense(128)	Dense(128)	Dense(128)	Dense(128)	Dense(128)	Dense(128)
Activation	relu	relu	relu	relu	relu	relu	relu
	Dropout (0.25)	Dropout (0.25)	Dropout (0.25)	Dropout (0.25)	Dropout (0.25)	Dropout (0.25)	Dropout (0.25)
Output layer	Dense (output = 1)	Dense (output = 1)	Dense (output = 1)	Dense (output = 1)	Dense (output = 1)	Dense (output = 1)	Dense (output = 1)
Activation	softmax	softmax	softmax	softmax	softmax	softmax	softmax
Image aug	Yes	Yes	Yes	Yes	Yes	Yes	Yes

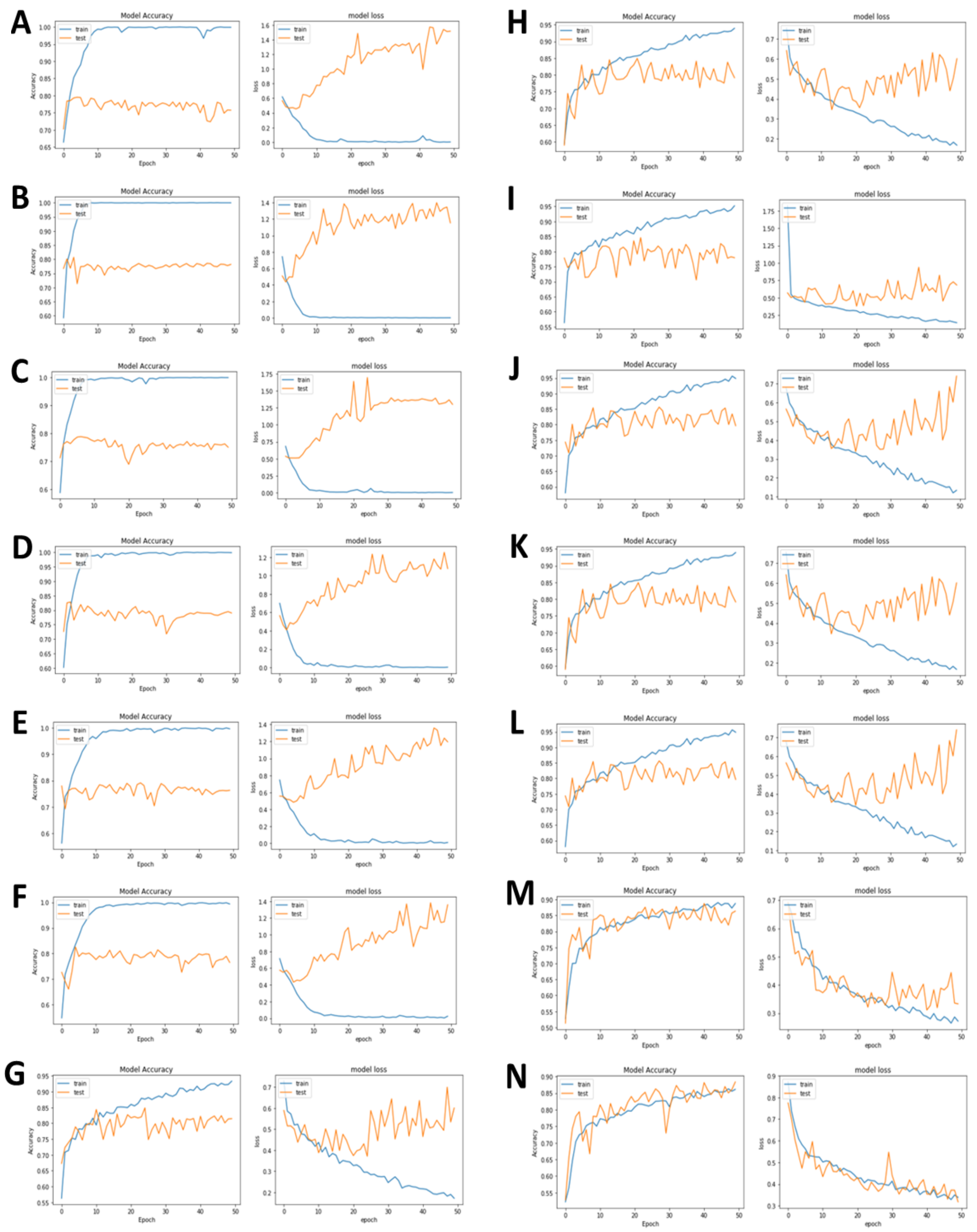


Figure 1. Training (blue line) and validation (orange line) accuracy (left panel) and loss (right panel) for the CNN model.

Table 3. Performance of various CNN models.

	CNN Network Configuration						
Performance	A	B	C	D	E	F	G
Train loss	0.0033	0.0013	0.0027	0.0036	0.0112	0.0244	0.1711
Train acc	0.9992	0.9996	0.9996	0.9988	0.9964	0.9936	0.9339
Val loss	1.5149	1.1535	1.3016	1.0771	1.1949	1.385	0.6003
Val acc	0.7577	0.7817	0.7507	0.7901	0.7628	0.7655	0.8147

Table 4. Performance of various CNN models.

	CNN Network Configuration						
Performance	H	I	J	K	L	M	N
Train loss	0.1679	0.1416	0.1327	0.1506	0.2614	0.2729	0.3387
Train acc	0.9399	0.9527	0.9491	0.9407	0.8878	0.8865	0.8613
Val loss	0.6016	0.6843	0.7411	0.4756	0.3266	0.3332	0.3176
Val acc	0.7916	0.7797	0.7966	0.8225	0.8654	0.8634	0.8833

We also checked to determine the performance of the CNN classifier for predicting the classes for images that it had not previously seen. Figure 2 shows predictions made by CNN network N for a set of random images taken from the test set. Overall, the CNN model performed well and was unsure for only one of the seven images.

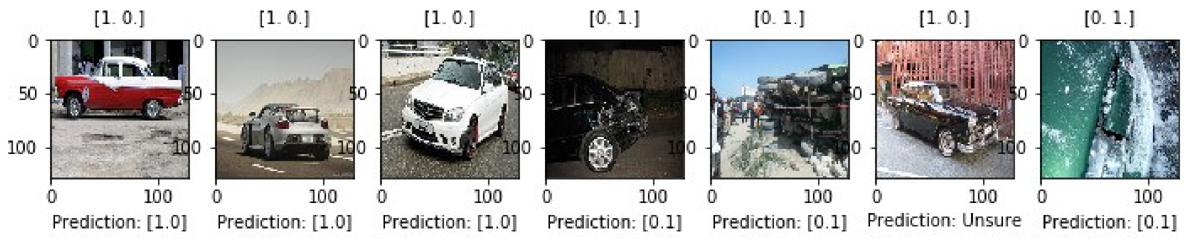


Figure 2. CNN model predictions for random images from the car dataset. The original classifications are shown above each image while predicted classification are below.

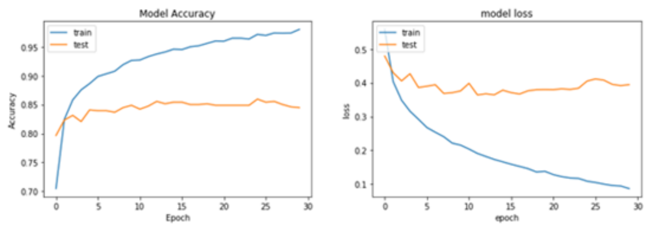
Transfer Learning

It has been shown that pre-trained networks can be utilized for deep learning on small datasets (also known as transfer learning). To test this, we used the large convolutional network VGG16 which was trained on the ImageNet dataset (1.4 million labelled images with 1000 different classes). There are 2 methods for leveraging pre-trained networks – feature extraction and fine-tuning.

Feature Extraction – Without data augmentation

We initially applied feature extraction, which uses representations learned by a previous network to extract meaningful features from our car dataset, without data augmentation (Fig. 3). The CNN model consisted of the convolutional base of the VGG16 model (containing the convolutional and pooling layers), which is the most generic and re-usable portion of the model, followed by a new densely connected classifier. The images were run through the convolution base of the VGG16 model using the weights previously acquired for this model and trained on the new densely connected part of the classifier (model F1). The model reached a validation accuracy of 84%, which was nearly was as good as model N (mentioned previously), however the model also exhibited overfitting (Fig. 3 & Table 5).

Figure 3. Transfer Learning with feature selection and no data augmentation. Training (blue line) and validation (orange line) accuracy (left panel) and loss (right panel) for the CNN model.



Feature Extraction – With data augmentation

To attempt to decrease overfitting we re-ran the feature extraction CNN model with data augmentation (i.e. rescale, shear, zoom and horizontal flip) (models F2). This resulted in a slight increase in the validation accuracy to 86% although overfitting was still observed (Fig. 4A). Increasing the number of augmentation parameters to include rotation, wide shift and height shift (model F3) however lead to a significant decrease in overfitting with a similar degree of accuracy (84%) (Fig. 4B & Table 5).

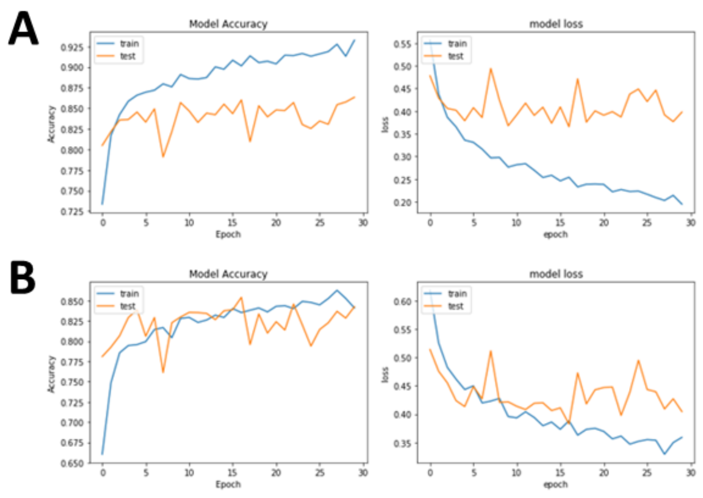


Figure 4. Transfer Learning with feature selection and data augmentation. (A) Data augmentation functions: (A) rescale = 1/255, shear = 0.2, zoom = 0.2, horizontal flip and (B) rescale = 1/255, shear = 0.2, zoom = 0.2, horizontal flip, rotation = 0.2, wide shift = 0.2, height shift = 0.2. Training (blue line) and validation (orange line) accuracy (left panel) and loss (right panel) for the CNN model.

Table 5. Performance of transfer trained CNN model using feature extraction.

	CNN Network Configuration		
Performance	F1	F2	F3
Train loss	0.0869	0.1974	0.3367
Train acc	0.9799	0.9313	0.8596
Val loss	0.3947	0.3978	0.3954
Val acc	0.8447	0.8633	0.8408

Fine Tuning

We were also interested in determining whether fine-tuning, another method for transferring the knowledge gained by a pre-trained network to a relatively small dataset, would improve performance. Unfreezing the final six convolutional layers of the VGG16 network did not lead to a significant improvement in the performance of the CNN model (Table 6).

Table 6. Performance of fine-tuned CNN model where a different number of convolutional layers of the VGG16 convolutional base have been frozen.

Performance	**CNN Network Configuration**				
	Conv4-1	Conv4-2	Conv5-1	Conv5-2	Conv5-3
Train loss	0.0069	0.0061	0.0080	0.0118	0.0284
Train acc	0.9968	0.9984	0.9978	0.9962	0.9928
Val loss	1.0976	0.7970	0.8805	0.6764	0.4565
Val acc	0.8601	0.8883	0.8601	0.8646	0.8716

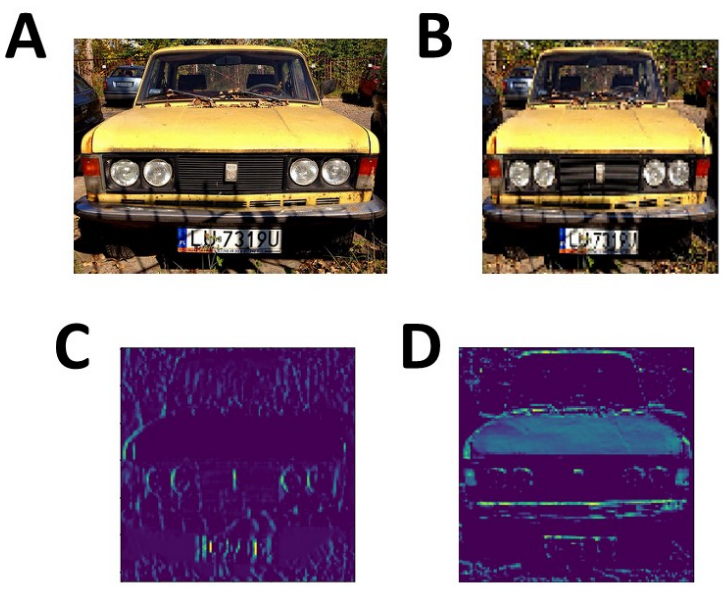
Visualization of Activation Layers

To get an idea of how the images were processed by the CNN network that we had created we plotted the activations, which essentially defined the feature maps generated by individual filters within each layer.

The first layer appeared to be retaining the full shape of the car wherein some filters retain portions of the outline of the car (Fig. 4C) while others retain surfaces (Fig. 4D). Several filters are not activated and left blank. Overall, the first convolutional layer appeared to retain a significant amount of information from the image.

Activations for layers became more abstract and less interpretable when proceeding deeper into the network (Fig. 5). The deeper layers appeared to encode higher level information such as borders, corners and angles. These layers likely provide less information about the visual content of the image and more information related to the class of the object in the image.

Figure 4. Activation map for CNN model. (A) Original and (B) processed image read into the network. (C) & (D) feature maps for different filters of a convolutional layer of the CNN model.



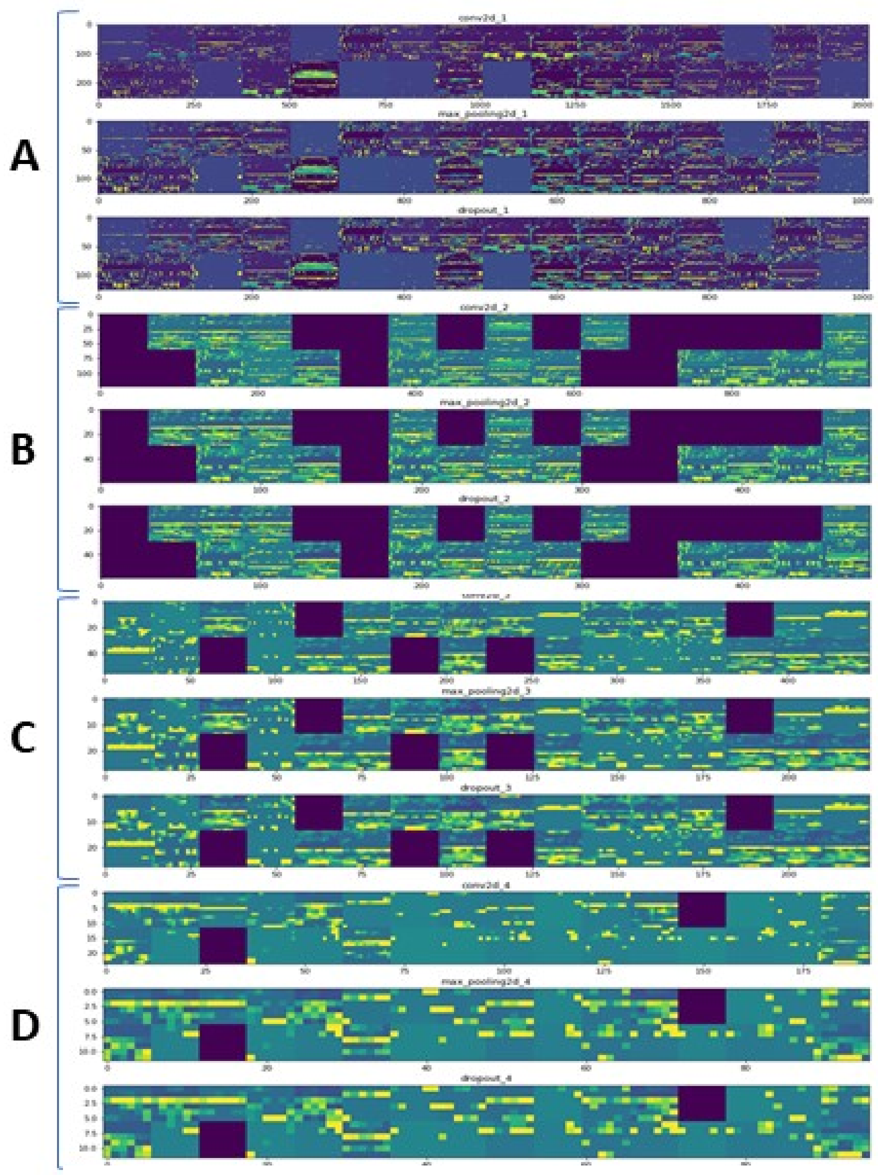


Figure 5. Feature maps for various convolutional layers of CNN model N. (A - D) Each layer is comprised of convolutional, maximum pooling and

Conclusions

The CNN network (model N) generated in these studies performed well with reasonably good accuracy and correctly predicting 6 out 7 random images taken from a test dataset. Due to difficulties in getting the images into a suitable format on the Google Colab server it was not possible to calculate additional measures of performance (e.g. confusion matrix, F1 score, precision and recall). Transfer learning using feature extract and utilizing the VG16 convolutional network did not perform well due to overfitting. Fine tuning produced a better result, similar to that of the generated CNN network (model N), with significantly less overfitting. Examination the feature maps suggested that this CNN network (model N) effectively captured information from images within the training dataset.

References

[1] F. Chollet, *Deep Learning with Python*. Manning Publications, 2017.

[2] C.-F. Blog, "Building powerful image classification models using very little data," *Keras Blog,* 2016.

[3] K. Simonyan and Z.-A. preprint arXiv, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556,* 2014.

[4] D. P. Kingma and B.-J. preprint arXiv, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980,* 2014.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural …,* 2012.

Appendices

Python code for model generation, transfer learning and feature map visualization.

CNN_Cars_train_predict_final2

October 18, 2019

#Deep Learning Model to Recognize Vehicle Accidents

[38]:
```
from google.colab import drive, files
drive.mount('/content/gdrive')
gdrive_path = 'gdrive/My\ Drive/Colab\ Notebooks/'
!rsync -ah --progress\
    {gdrive_path}cars_dl/*.zip\
    {gdrive_path}model_weights_final*.h5\
    {gdrive_path}model_structure_final*.json\
    '/content'
!unzip -qo '*.zip'
!rm *.zip
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call
drive.mount("/content/gdrive", force_remount=True).
sending incremental file list
rsync: link_stat "/content/gdrive/My Drive/Colab
Notebooks/model_weights_final*.h5" failed: No such file or directory (2)
rsync: link_stat "/content/gdrive/My Drive/Colab
Notebooks/model_structure_final*.json" failed: No such file or directory (2)
cars-20191002T050118Z-001.zip
        490.68M 100%  142.96MB/s    0:00:03 (xfr#1, to-chk=0/1)
rsync error: some files/attrs were not transferred (see previous errors) (code
23) at main.c(1196) [sender=3.1.2]
```

##Import Keras library and packages

[2]:
```
# Importing the Keras libraries and packages
from sklearn.metrics import classification_report, confusion_matrix
from keras.layers import Convolution2D, MaxPooling2D, Dropout
from keras.models import Sequential
from keras.layers import Flatten
from keras.layers import Dense
from keras import backend as K
from keras.preprocessing import image
from keras import regularizers
from PIL import Image
from pathlib import Path
```

```
import numpy as np
import matplotlib.pyplot as plt
import os
```

```
Using TensorFlow backend.
```

##Set parameters

[0]:
```
epochs = 50 #@param {type:"number"}
validation_steps = 32 #@param {type:"number"}
img_height = 128 #@param {type:"integer"}
img_width = 128 #@param {type:"integer"}
batch_size = 32 #@param {type:"number"}
```

##Set data filepaths

[4]:
```
trainingCarImages = os.listdir('cars/training_set/car/')
print ("Number of Training car images - ",str(len(trainingCarImages)))

trainingCrashImages = os.listdir('cars/training_set/crash/')
print ("Number of Training crash images - ",str(len(trainingCrashImages)))

validCarImages = os.listdir('cars/val_set/car/')
print ("Number of Validation car images - ",str(len(validCarImages)))

validCrashImages = os.listdir('cars/val_set/crash/')
print ("Number of Validation crash images - ",str(len(validCrashImages)))

testCarImages = os.listdir('cars/test_set/car/')
print ("Number of Test car images - ",str(len(testCarImages)))

testCrashImages = os.listdir('cars/test_set/crash/')
print ("Number of Test crash images - ",str(len(testCrashImages)))
```

```
Number of Training car images -  1445
Number of Training crash images -  1445
Number of Validation car images -  375
Number of Validation crash images -  372
Number of Test car images -  129
Number of Test crash images -  131
```

##Print image from training folder

[5]:
```
trainFilename = 'cars/training_set/car/'+trainingCarImages[2]
dimage = Image.open(trainFilename)
dimage
```

[5]:

##Print image from validation folder

[6]:
```
validFilename = 'cars/val_set/car/'+validCarImages[5]
dimage = Image.open(validFilename)
dimage
```

[6]:

##Print image from test folder

[7]:
```
testFilename = 'cars/test_set/car/'+ testCarImages[30]
dimage = Image.open(testFilename)
dimage
```

[7]:

###Determine image dimensions

[8]:
```
train_data_dir = 'cars/training_set/'
validation_data_dir = 'cars/val_set/'
test_data_dir = 'cars/test_set/'
nb_train_samples = trainingCarImages + trainingCrashImages
nb_validation_samples = validCarImages + validCrashImages
nb_test_samples = testCarImages + testCrashImages

if K.image_data_format() == 'channels_first':
    input_shape = (3, img_width, img_height)
else:
    input_shape = (img_width, img_height, 3)

print(input_shape)
```

```
(128, 128, 3)
```

[0]:
```
#Method to print several images in a single row
def plots(img, figsize=(12,6), rows = 1, titles = 1):
    if type(img[0]) is np.ndarray:
        img = np.array(img).astype(np.float_)
        if (img.shape[-1] != 3):
            img = img.transpose((0, 2, 3, 1))
    f =plt.figure(figsize = figsize)
    cols = 7//rows if (len(img) % 2 == 0) else len(img)//rows + 1
    for i in range(cols):
        sp = f.add_subplot(rows, cols, i+1)
        sp.axis('Off')
        if titles is not None:
            sp.set_title(titles[i], fontsize = 10)
        plt.imshow(img[i], interpolation = None if np.interp else 'none')
```

##Create CNN model

[10]:
```
# Initialising the CNN
classifier = Sequential()

# Layer 1 - convolution
classifier.add(Convolution2D(32, (3, 3),
              input_shape = (img_width, img_height, 3),
              activation = 'relu'))                          # Convolution
classifier.add(MaxPooling2D(pool_size = (2, 2)))            # MaxPooling
classifier.add(Dropout(0.25))                               # Dropout


#Add another convolutional layer
# classifier.add(Convolution2D(32, 3, 3,
#                activation = 'relu'))
# classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Layer 2 - convolution
classifier.add(Convolution2D(32, (3, 3),
              activation = 'relu',
              kernel_regularizer=regularizers.l2(0.01)))  # Convolution layer
classifier.add(MaxPooling2D(pool_size = (2, 2)))            # MaxPooling
classifier.add(Dropout(0.25))                               # Dropout

# Layer 3 - convolution
classifier.add(Convolution2D(32, (3, 3),
              activation = 'relu'))                         # Convolution layer
classifier.add(MaxPooling2D(pool_size = (2, 2)))            # MaxPooling
classifier.add(Dropout(0.25))                               # Dropout

# Layer 4 - convolution
classifier.add(Convolution2D(32, (3, 3),
              activation = 'relu'))                         # Convolution layer
classifier.add(MaxPooling2D(pool_size = (2, 2)))            # MaxPooling
classifier.add(Dropout(0.25))                               # Dropout

classifier.add(Flatten())                                   # Flattening

# Fully connected layer
classifier.add(Dense(output_dim = 256,
```

```python
                     activation = 'relu'))                # Dense layer
classifier.add(Dropout(0.25))                               # Dropout
classifier.add(Dense(output_dim = 2,
                     activation = 'softmax'))             # Output

classifier.summary()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:66: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4432: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4267: The name tf.nn.max_pool is
deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:148: The name
tf.placeholder_with_default is deprecated. Please use
tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3733: calling dropout (from
tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed
in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 -
keep_prob`.
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 126, 126, 32)      896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 63, 63, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 61, 61, 32)        9248
_________________________________________________________________
```

```
max_pooling2d_2 (MaxPooling2 (None, 30, 30, 32)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 30, 30, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 32)        9248
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
dropout_3 (Dropout)          (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 12, 12, 32)        9248
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 6, 6, 32)          0
_________________________________________________________________
dropout_4 (Dropout)          (None, 6, 6, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               295168
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 514
=================================================================
Total params: 324,322
Trainable params: 324,322
Non-trainable params: 0
_________________________________________________________________

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:39: UserWarning:
Update your `Dense` call to the Keras 2 API: `Dense(activation="relu",
units=256)`
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:42: UserWarning:
Update your `Dense` call to the Keras 2 API: `Dense(activation="softmax",
units=2)`
```

##Compiling the CNN

[11]:
```
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics =␣
↪['accuracy'])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated.
Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3657: The name tf.log is
deprecated. Please use tf.math.log instead.
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow_core/python/ops/nn_impl.py:183: where (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

##Data standardization and augmentation of training data

[12]:
```python
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                    shear_range = 0.2,
                                    zoom_range = 0.2,
                                    horizontal_flip = True
                                   )

training_set = train_datagen.flow_from_directory(train_data_dir,
                                                 target_size = (img_width,␣
 ↪img_height),
                                                 batch_size = batch_size,
                                                 class_mode = 'categorical',
                                                 shuffle=True,
                                                 seed=42)

test_datagen = ImageDataGenerator(rescale = 1./255)

val_set = test_datagen.flow_from_directory(validation_data_dir,
                                            target_size = (img_width,␣
 ↪img_height),
                                            batch_size = batch_size,
                                            class_mode = 'categorical',
                                            shuffle=True,
                                            seed=42)

test_set = test_datagen.flow_from_directory(test_data_dir,
                                            target_size = (img_width,␣
 ↪img_height),
                                            batch_size = batch_size,
                                            class_mode = 'categorical'
                                            )
```

```
Found 2890 images belonging to 2 classes.
Found 747 images belonging to 2 classes.
Found 260 images belonging to 2 classes.
```

##Print normalized and re-sized images from training set

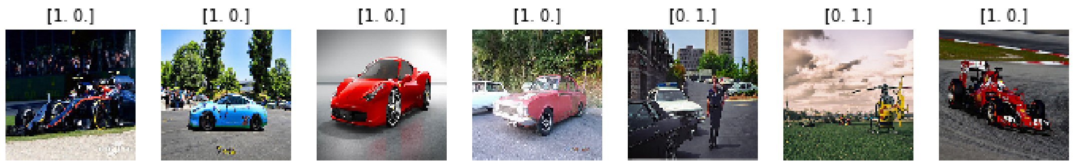
[13]:
```
imgs, labels = next(training_set)
plots(imgs, titles = labels)
```

```
/usr/local/lib/python3.6/dist-packages/matplotlib/text.py:1191: FutureWarning:
elementwise comparison failed; returning scalar instead, but in the future will
perform elementwise comparison
  if s != self._text:
```

##Print normalized and re-sized images from validation set

[14]:
```
imgs, labels = next(val_set)
plots(imgs, titles = labels)
```

```
/usr/local/lib/python3.6/dist-packages/matplotlib/text.py:1191: FutureWarning:
elementwise comparison failed; returning scalar instead, but in the future will
perform elementwise comparison
  if s != self._text:
```



[0]:
```
#Method to identify and remove corrupted images
def my_gen(gen):
    while True:
        try:
            imgs, labels = next(gen)
            yield imgs, labels
        except:
            pass
```

##Training the CNN model

[16]:
```
history = classifier.fit_generator(my_gen(training_set),
                                   steps_per_epoch = 78,
                                   epochs = epochs,
                                   validation_data = val_set,
                                   validation_steps = validation_steps)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:1033: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:1020: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3005: The name tf.Session is
deprecated. Please use tf.compat.v1.Session instead.

Epoch 1/50
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:190: The name
tf.get_default_session is deprecated. Please use
tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:197: The name tf.ConfigProto is
deprecated. Please use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:207: The name tf.global_variables
is deprecated. Please use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:216: The name
tf.is_variable_initialized is deprecated. Please use
tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:223: The name
tf.variables_initializer is deprecated. Please use
tf.compat.v1.variables_initializer instead.


/usr/local/lib/python3.6/dist-packages/PIL/Image.py:914: UserWarning: Palette
images with Transparency   expressed in bytes should be converted to RGBA images
  'to RGBA images')

78/78 [==============================] - 44s 565ms/step - loss: 0.8839 - acc:
0.5457 - val_loss: 0.7986 - val_acc: 0.4965
Epoch 2/50
78/78 [==============================] - 37s 480ms/step - loss: 0.7383 - acc:
```

```
0.5880 - val_loss: 0.6595 - val_acc: 0.6929
Epoch 3/50
78/78 [==============================] - 36s 461ms/step - loss: 0.6470 - acc:
0.6722 - val_loss: 0.5711 - val_acc: 0.7393
Epoch 4/50
78/78 [==============================] - 36s 464ms/step - loss: 0.6129 - acc:
0.7078 - val_loss: 0.5382 - val_acc: 0.7866
Epoch 5/50
78/78 [==============================] - 37s 477ms/step - loss: 0.5806 - acc:
0.7273 - val_loss: 0.4868 - val_acc: 0.7996
Epoch 6/50
78/78 [==============================] - 35s 452ms/step - loss: 0.5677 - acc:
0.7352 - val_loss: 0.5452 - val_acc: 0.6996
Epoch 7/50
78/78 [==============================] - 36s 465ms/step - loss: 0.5351 - acc:
0.7578 - val_loss: 0.4955 - val_acc: 0.7587
Epoch 8/50
78/78 [==============================] - 36s 460ms/step - loss: 0.5256 - acc:
0.7520 - val_loss: 0.5604 - val_acc: 0.6949
Epoch 9/50
78/78 [==============================] - 37s 468ms/step - loss: 0.5122 - acc:
0.7719 - val_loss: 0.5059 - val_acc: 0.7597
Epoch 10/50
78/78 [==============================] - 35s 454ms/step - loss: 0.5200 - acc:
0.7603 - val_loss: 0.4846 - val_acc: 0.7996
Epoch 11/50
78/78 [==============================] - 38s 486ms/step - loss: 0.5041 - acc:
0.7715 - val_loss: 0.4388 - val_acc: 0.8275
Epoch 12/50
78/78 [==============================] - 33s 427ms/step - loss: 0.5011 - acc:
0.7704 - val_loss: 0.4565 - val_acc: 0.8075
Epoch 13/50
78/78 [==============================] - 38s 491ms/step - loss: 0.4867 - acc:
0.7796 - val_loss: 0.5492 - val_acc: 0.7188
Epoch 14/50
78/78 [==============================] - 37s 476ms/step - loss: 0.4885 - acc:
0.7757 - val_loss: 0.4668 - val_acc: 0.8156
Epoch 15/50
78/78 [==============================] - 36s 467ms/step - loss: 0.4842 - acc:
0.7771 - val_loss: 0.4279 - val_acc: 0.8289
Epoch 16/50
78/78 [==============================] - 34s 441ms/step - loss: 0.4557 - acc:
0.7926 - val_loss: 0.4836 - val_acc: 0.7777
Epoch 17/50
78/78 [==============================] - 37s 477ms/step - loss: 0.4505 - acc:
0.7978 - val_loss: 0.4378 - val_acc: 0.8215
Epoch 18/50
78/78 [==============================] - 37s 480ms/step - loss: 0.4601 - acc:
```

```
0.7850 - val_loss: 0.4647 - val_acc: 0.7953
Epoch 19/50
78/78 [==============================] - 36s 466ms/step - loss: 0.4516 - acc:
0.7938 - val_loss: 0.4147 - val_acc: 0.8415
Epoch 20/50
78/78 [==============================] - 35s 450ms/step - loss: 0.4318 - acc:
0.8079 - val_loss: 0.4132 - val_acc: 0.8435
Epoch 21/50
78/78 [==============================] - 38s 483ms/step - loss: 0.4201 - acc:
0.8188 - val_loss: 0.3866 - val_acc: 0.8595
Epoch 22/50
78/78 [==============================] - 36s 458ms/step - loss: 0.4187 - acc:
0.8165 - val_loss: 0.3841 - val_acc: 0.8385
Epoch 23/50
78/78 [==============================] - 34s 440ms/step - loss: 0.4033 - acc:
0.8288 - val_loss: 0.4196 - val_acc: 0.8285
Epoch 24/50
78/78 [==============================] - 37s 477ms/step - loss: 0.4152 - acc:
0.8196 - val_loss: 0.3669 - val_acc: 0.8564
Epoch 25/50
78/78 [==============================] - 35s 450ms/step - loss: 0.4127 - acc:
0.8180 - val_loss: 0.4473 - val_acc: 0.8106
Epoch 26/50
78/78 [==============================] - 37s 477ms/step - loss: 0.4009 - acc:
0.8241 - val_loss: 0.3688 - val_acc: 0.8564
Epoch 27/50
78/78 [==============================] - 36s 468ms/step - loss: 0.3896 - acc:
0.8244 - val_loss: 0.3797 - val_acc: 0.8483
Epoch 28/50
78/78 [==============================] - 35s 452ms/step - loss: 0.3957 - acc:
0.8284 - val_loss: 0.3968 - val_acc: 0.8415
Epoch 29/50
78/78 [==============================] - 37s 474ms/step - loss: 0.3895 - acc:
0.8281 - val_loss: 0.3861 - val_acc: 0.8385
Epoch 30/50
78/78 [==============================] - 38s 491ms/step - loss: 0.3844 - acc:
0.8380 - val_loss: 0.4305 - val_acc: 0.8269
Epoch 31/50
78/78 [==============================] - 36s 461ms/step - loss: 0.3882 - acc:
0.8350 - val_loss: 0.5100 - val_acc: 0.7717
Epoch 32/50
78/78 [==============================] - 37s 469ms/step - loss: 0.3662 - acc:
0.8489 - val_loss: 0.4053 - val_acc: 0.8345
Epoch 33/50
78/78 [==============================] - 37s 471ms/step - loss: 0.3858 - acc:
0.8343 - val_loss: 0.3662 - val_acc: 0.8493
Epoch 34/50
78/78 [==============================] - 36s 464ms/step - loss: 0.3728 - acc:
```

```
0.8469 - val_loss: 0.3675 - val_acc: 0.8514
Epoch 35/50
78/78 [==============================] - 35s 454ms/step - loss: 0.3680 - acc:
0.8365 - val_loss: 0.3592 - val_acc: 0.8754
Epoch 36/50
78/78 [==============================] - 40s 515ms/step - loss: 0.3401 - acc:
0.8628 - val_loss: 0.3897 - val_acc: 0.8401
Epoch 37/50
78/78 [==============================] - 36s 458ms/step - loss: 0.3602 - acc:
0.8457 - val_loss: 0.3935 - val_acc: 0.8405
Epoch 38/50
78/78 [==============================] - 38s 482ms/step - loss: 0.3806 - acc:
0.8393 - val_loss: 0.3557 - val_acc: 0.8654
Epoch 39/50
78/78 [==============================] - 36s 464ms/step - loss: 0.3625 - acc:
0.8557 - val_loss: 0.4326 - val_acc: 0.8177
Epoch 40/50
78/78 [==============================] - 36s 458ms/step - loss: 0.3508 - acc:
0.8512 - val_loss: 0.4766 - val_acc: 0.7936
Epoch 41/50
78/78 [==============================] - 36s 465ms/step - loss: 0.3418 - acc:
0.8574 - val_loss: 0.3505 - val_acc: 0.8744
Epoch 42/50
78/78 [==============================] - 38s 489ms/step - loss: 0.3452 - acc:
0.8604 - val_loss: 0.3599 - val_acc: 0.8605
Epoch 43/50
78/78 [==============================] - 35s 454ms/step - loss: 0.3495 - acc:
0.8549 - val_loss: 0.3676 - val_acc: 0.8534
Epoch 44/50
78/78 [==============================] - 36s 464ms/step - loss: 0.3348 - acc:
0.8612 - val_loss: 0.3227 - val_acc: 0.8824
Epoch 45/50
78/78 [==============================] - 38s 482ms/step - loss: 0.3367 - acc:
0.8601 - val_loss: 0.3496 - val_acc: 0.8737
Epoch 46/50
78/78 [==============================] - 37s 468ms/step - loss: 0.3280 - acc:
0.8648 - val_loss: 0.3272 - val_acc: 0.8764
Epoch 47/50
78/78 [==============================] - 38s 482ms/step - loss: 0.3386 - acc:
0.8582 - val_loss: 0.3796 - val_acc: 0.8345
Epoch 48/50
78/78 [==============================] - 36s 465ms/step - loss: 0.3082 - acc:
0.8777 - val_loss: 0.3850 - val_acc: 0.8544
Epoch 49/50
78/78 [==============================] - 37s 469ms/step - loss: 0.3243 - acc:
0.8686 - val_loss: 0.3641 - val_acc: 0.8614
Epoch 50/50
78/78 [==============================] - 36s 465ms/step - loss: 0.3261 - acc:
```

```
0.8755 - val_loss: 0.3458 - val_acc: 0.8524
```

[33]:
```
# list all data in history
print(history.history.keys())
```

```
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```

[63]:
```
!ls '/My Drive/Colab Notebooks'
```

```
ls: cannot access '/My Drive/Colab Notebooks': No such file or directory
```

[64]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

0.1 Save neural network structure

[66]:
```
from pathlib import Path
import os
model_struct_final = classifier.to_json()
f = Path("/content/drive/My Drive/Colab Notebooks/model_struct_final.json")
f.write_text(model_struct_final)
```

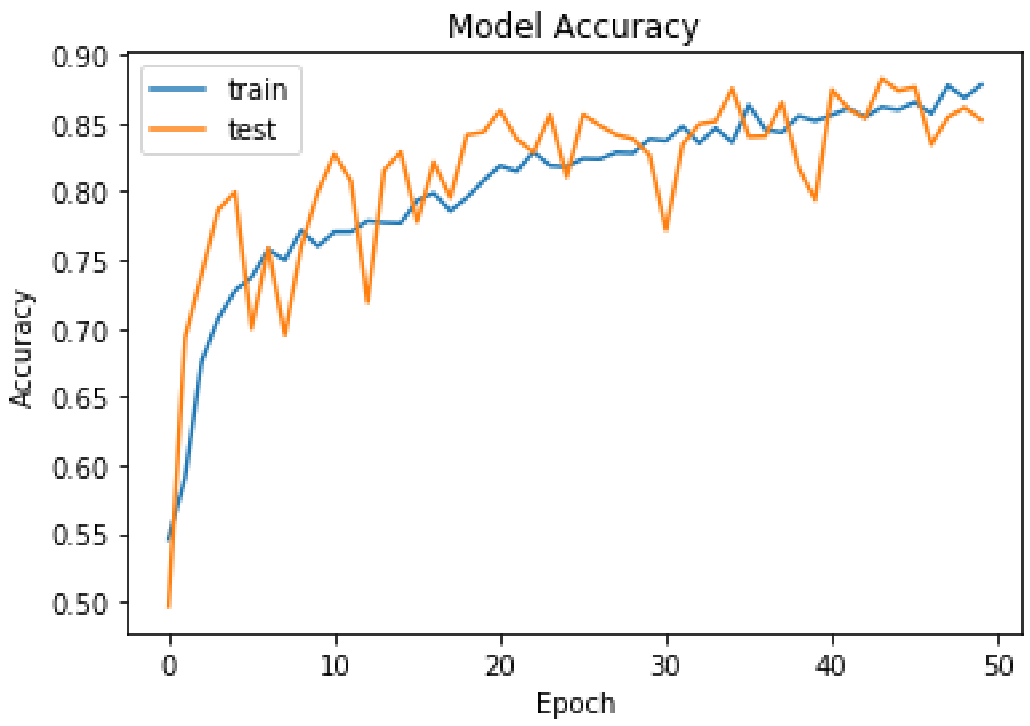
[66]:
```
5328
```

0.2 Save neural network's trained weights

[0]:
```
# Save neural network's trained weights
classifier.save_weights("/content/drive/My Drive/Colab Notebooks/
↪model_weights_final.h5")
```

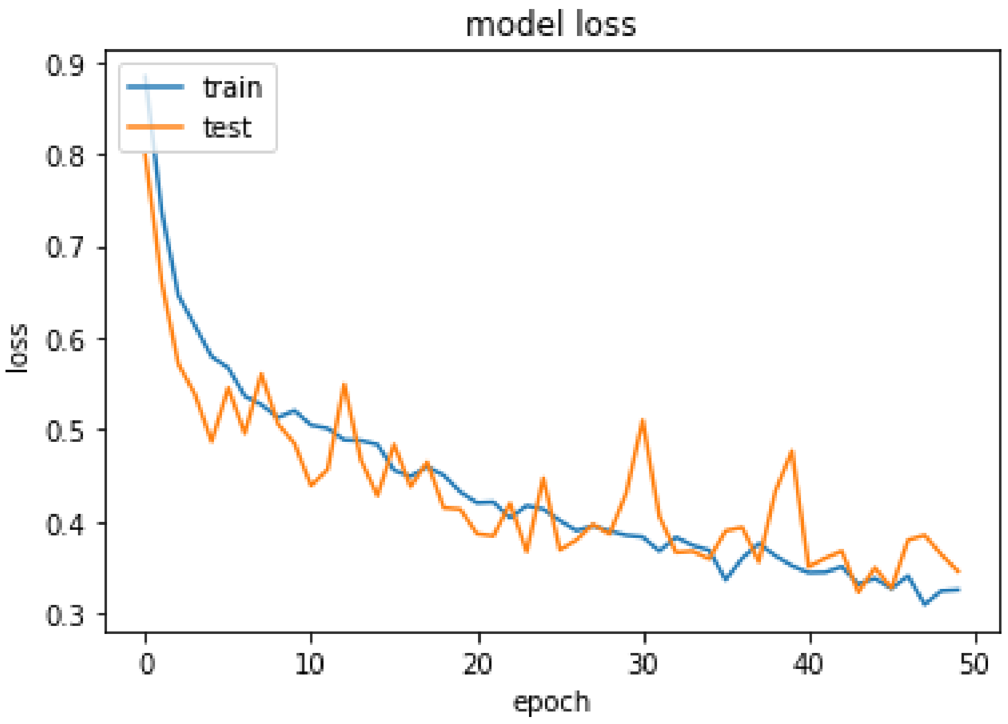
##Plot training and validation accuracy per epoch

[30]:
```
import matplotlib.pyplot as plt
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



##Plot training and validation loss per epoch

[21]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



[22]:
```
training_set.class_indices
```

[22]:
```
{'car': 0, 'crash': 1}
```

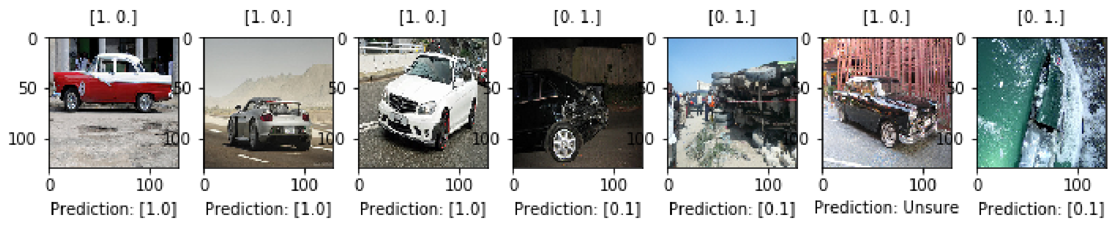
##Prediction

[23]:
```
#Method to print several images in a single row
def plots(img, figsize=(12,6), rows = 1, titles = 1):

  if type(img[0]) is np.ndarray:
    img = np.array(img).astype(np.float_)
  if (img.shape[-1] != 3):
    img = img.transpose((0, 2, 3, 1))
  f =plt.figure(figsize = figsize)
  cols = 7//rows if (len(img) % 2 == 0) else len(img)//rows + 1

  for i in range(cols):
    sp = f.add_subplot(rows, cols, i+1)
    sp.axis('On')

    result = classifier.predict(img)
    print(result[i][0])
    test_set.class_indices
    if result[i][0] < 0.3:
      prediction = '[0.1]'
    elif result[i][0] > 0.7:
      prediction = '[1.0]'
    else:
      prediction = 'Unsure'

    if titles is not None:
      sp.set_title(titles[i], fontsize = 10, pad = 10)
      sp.set_xlabel('Prediction: ' + prediction)
    plt.imshow(img[i], interpolation = None if np.interp else 'none')


imgs, labels = next(test_set)
plots(imgs, titles = labels)
```

```
0.9375169
0.8874122
0.94651306
0.26875815
0.009791187
```

```
/usr/local/lib/python3.6/dist-packages/matplotlib/text.py:1191: FutureWarning:
elementwise comparison failed; returning scalar instead, but in the future will
perform elementwise comparison
  if s != self._text:
```

```
0.48972613
0.056924686
```



###Process images and store in different folder

[0]:
```
import os, shutil

# The directory where we will store our cars dataset
base_dir = 'cars_pro/'
os.mkdir(base_dir)

# Directories for our training, validation and test splits
test_dir = os.path.join(base_dir, 'test_set_pro')
os.mkdir(test_dir)
```

[25]:
```
from keras.preprocessing.image import ImageDataGenerator
from pathlib import Path


test_data_dir = 'cars/test_set/'

# All images will be rescaled by 1./255
test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=(128, 128),
    batch_size=10,
    class_mode = 'categorical')

print("OK")
for data_batch, labels_batch in test_generator:
  print('data batch shape:', data_batch.shape)
  print('labels batch shape:', labels_batch.shape)
  break
```

```
Found 260 images belonging to 2 classes.
OK
data batch shape: (10, 128, 128, 3)
labels batch shape: (10, 2)

Exception ignored in: <generator object my_gen at 0x7f0d5bf5daf0>
RuntimeError: generator ignored GeneratorExit
```

[0]:
```
# This is module with image preprocessing utilities
from keras.preprocessing import image
import matplotlib.pyplot as plt
from scipy import ndarray
import skimage as sk
from skimage import transform
from skimage import util

datagen = ImageDataGenerator(
    rescale=1./255
    )

test_data_dir = 'cars/test_set/car/'
fnames = [os.path.join(test_data_dir, fname) for fname in os.
 ↪listdir(test_data_dir)]

for i in range(1, len(fnames), 1):
  # We pick one image to "augment"
  img_path = fnames[i]

  # Read the image and resize it
  img = image.load_img(img_path, target_size=(128, 128))

  # Convert it to a Numpy array with shape (128, 128, 3)
  x = image.img_to_array(img)

  # Reshape it to (1, 128, 128, 3)
  x = x.reshape((1,) + x.shape)

  car_pro_dir = 'cars_pro/'
  img.save(os.path.join(car_pro_dir, 'car' + str(i) + '.jpg'))
```

##Confusion matrix and classification report

[27]:
```
#Confution Matrix and Classification Report
from sklearn.metrics import classification_report, confusion_matrix
Y_pred = classifier.predict_classes(car_pro_dir)
type(Y_pred)
y_pred = np.argmax(Y_pred, axis=1)

print('Confusion Matrix')

print(confusion_matrix(y_true = car_pro_dir.classes, y_pred = y_pred))
#confusion_matrix = metrics.confusion_matrix(y_true=y_true_labels,␣
 ↪y_pred=y_pred_labels)  # shape=(12, 12)


print('Classification Report')

target_names = ['Yes', 'No']

print(classification_report(y_true = car_pro_dir.classes, y_pred = y_pred,␣
 ↪target_names = target_names))
```

```
      ␣
↪---------------------------------------------------------------------------

      AttributeError                            Traceback (most recent call␣
↪last)

      <ipython-input-27-cad059fa386e> in <module>()
        1 from sklearn.metrics import classification_report, confusion_matrix
  ----> 2 Y_pred = classifier.predict_classes(car_pro_dir)
        3 type(Y_pred)
        4 y_pred = np.argmax(Y_pred, axis=1)
      5

    /usr/local/lib/python3.6/dist-packages/keras/engine/sequential.py in␣
↪predict_classes(self, x, batch_size, verbose)
    266             A numpy array of class predictions.
    267         """
--> 268         proba = self.predict(x, batch_size=batch_size,␣
↪verbose=verbose)
    269         if proba.shape[-1] > 1:
    270             return proba.argmax(axis=-1)


    /usr/local/lib/python3.6/dist-packages/keras/engine/training.py in␣
↪predict(self, x, batch_size, verbose, steps, callbacks, max_queue_size,␣
↪workers, use_multiprocessing)
   1378
   1379         # Case 2: Symbolic tensors or Numpy array-like.
-> 1380         x, _, _ = self._standardize_user_data(x)
   1381         if self.stateful:
   1382             if x[0].shape[0] > batch_size and x[0].shape[0] %␣
↪batch_size != 0:


    /usr/local/lib/python3.6/dist-packages/keras/engine/training.py in␣
↪_standardize_user_data(self, x, y, sample_weight, class_weight,␣
↪check_array_lengths, batch_size)
    755             feed_input_shapes,
    756             check_batch_axis=False,  # Don't enforce the batch size.
--> 757             exception_prefix='input')
    758
    759         if y is not None:


    /usr/local/lib/python3.6/dist-packages/keras/engine/training_utils.py in␣
↪standardize_input_data(data, names, shapes, check_batch_axis, exception_prefix)
     93         data = data.values if data.__class__.__name__ == 'DataFrame'␣
↪else data
     94         data = [data]
---> 95     data = [standardize_single_array(x) for x in data]
     96
     97     if len(data) != len(names):


    /usr/local/lib/python3.6/dist-packages/keras/engine/training_utils.py in␣
↪<listcomp>(.0)
```

```
     93         data = data.values if data.__class__.__name__ == 'DataFrame'␣
↪else data
     94         data = [data]
---> 95     data = [standardize_single_array(x) for x in data]
     96 
     97     if len(data) != len(names):


    /usr/local/lib/python3.6/dist-packages/keras/engine/training_utils.py in␣
↪standardize_single_array(x)
     28                 'Got tensor with shape: %s' % str(shape))
     29         return x
---> 30     elif x.ndim == 1:
     31         x = np.expand_dims(x, 1)
     32     return x


    AttributeError: 'str' object has no attribute 'ndim'
```

cars_transfer_learning_feat_extract_without_data_aug

October 19, 2019

#CNN model of vehicle recognition - Feature extraction without data augmentation

[1]:
```
from google.colab import drive, files
drive.mount('/content/gdrive')
gdrive_path = 'gdrive/My\ Drive/Colab\ Notebooks/'
!rsync -ah --progress\
    {gdrive_path}cars_dl/*.zip\
    {gdrive_path}model_weights_final*.h5\
    {gdrive_path}model_struct_final*.json\
    '/content'
!unzip -qo '*.zip'
!rm *.zip
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id
=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redire
ct_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2Fwww.googl
eapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive%2
0https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2
Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code

Enter your authorization code:
ůůůůůůůůůů
Mounted at /content/gdrive
sending incremental file list
cars-20191002T050118Z-001.zip
        490.68M 100%   72.09MB/s    0:00:06 (xfr#1, to-chk=2/3)
model_struct_final.json
          5.33K 100%    7.19kB/s    0:00:00 (xfr#2, to-chk=1/3)
model_weights_final.h5
          1.33M 100%    1.12MB/s    0:00:01 (xfr#3, to-chk=0/3)
```

[0]:
```
epochs = 50 #@param {type:"number"}
validation_steps = 32 #@param {type:"number"}
img_height = 128 #@param {type:"integer"}
img_width = 128 #@param {type:"integer"}
batch_size = 32 #@param {type:"number"}
```

##Example of using a pre-trained model (VGG16) as a classifier

[3]:
```python
# Load Keras' VGG16 model that was pre-trained against the ImageNet database
from keras.applications import VGG16
conv_base = VGG16(weights='imagenet',
include_top=False,
input_shape=(128, 128, 3))
```

```
Using TensorFlow backend.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:66: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4432: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4267: The name tf.nn.max_pool is
deprecated. Please use tf.nn.max_pool2d instead.

Downloading data from https://github.com/fchollet/deep-learning-
models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 3s 0us/step
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:190: The name
tf.get_default_session is deprecated. Please use
tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:197: The name tf.ConfigProto is
deprecated. Please use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:203: The name tf.Session is
deprecated. Please use tf.compat.v1.Session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:207: The name tf.global_variables
is deprecated. Please use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:216: The name
tf.is_variable_initialized is deprecated. Please use
tf.compat.v1.is_variable_initialized instead.
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:223: The name
tf.variables_initializer is deprecated. Please use
tf.compat.v1.variables_initializer instead.
```

[4]:
```
# Summary of Keras' VGG16 model that was pre-trained against the ImageNet␣
↪database
conv_base.summary()
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 128, 128, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 128, 128, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 128, 128, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 64, 64, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 64, 64, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 64, 64, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 32, 32, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 32, 32, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 32, 32, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 32, 32, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 16, 16, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 16, 16, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 8, 8, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
```

```
block5_conv2 (Conv2D)        (None, 8, 8, 512)        2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 8, 8, 512)        2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 4, 4, 512)        0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

##Number of images in the training, validation and test sets

[5]:
```
import os

trainingCarImages = os.listdir('cars/training_set/car/')
print ("Number of Training car images - ",str(len(trainingCarImages)))

trainingCrashImages = os.listdir('cars/training_set/crash/')
print ("Number of Training crash images - ",str(len(trainingCrashImages)))

validCarImages = os.listdir('cars/val_set/car/')
print ("Number of Validation car images - ",str(len(validCarImages)))

validCrashImages = os.listdir('cars/val_set/crash/')
print ("Number of Validation crash images - ",str(len(validCrashImages)))

testCarImages = os.listdir('cars/test_set/car/')
print ("Number of Test car images - ",str(len(testCarImages)))

testCrashImages = os.listdir('cars/test_set/crash/')
print ("Number of Test crash images - ",str(len(testCrashImages)))

train_data_dir = 'cars/training_set/'
validation_data_dir = 'cars/val_set/'
test_data_dir = 'cars/test_set/'
nb_train_samples = trainingCarImages + trainingCrashImages
nb_validation_samples = validCarImages + validCrashImages
nb_test_samples = testCarImages + testCrashImages
```

```
Number of Training car images -  1445
Number of Training crash images -  1445
Number of Validation car images -  375
Number of Validation crash images -  372
Number of Test car images -  129
Number of Test crash images -  131
```

[0]:
```
#Method to identify and remove corrupted images
def my_gen(gen):
    while True:
        try:
            imgs, labels = next(gen)
            yield imgs, labels
        except:
            pass
```

##Extract features using the pre-trained convolutional base

[7]:
```
import os
import numpy as np
from keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(rescale=1./255)
batch_size = 10

def extract_features(directory, sample_count):
  features = np.zeros(shape=(sample_count, 4, 4, 512))
  labels = np.zeros(shape=(sample_count))
  generator = datagen.flow_from_directory(
      directory,
      target_size=(128, 128),
      batch_size=batch_size,
      class_mode='binary')
  i = 0
  generator = my_gen(generator)
  for inputs_batch, labels_batch in generator:
    features_batch = conv_base.predict(inputs_batch)
    features[i * batch_size : (i + 1) * batch_size] = features_batch
    labels[i * batch_size : (i + 1) * batch_size] = labels_batch
    i += 1
    if i * batch_size >= sample_count:
      # Note that since generators yield data indefinitely in a loop,
      # we must `break` after every image has been seen once.
      break
  return features, labels

train_features, train_labels = extract_features(train_data_dir, 2890)
validation_features, validation_labels = extract_features(validation_data_dir,␣
 ↪747)
test_features, test_labels = extract_features(test_data_dir, 260)
```

```
Found 2890 images belonging to 2 classes.

/usr/local/lib/python3.6/dist-packages/PIL/Image.py:914: UserWarning: Palette
images with Transparency   expressed in bytes should be converted to RGBA images
  'to RGBA images')
```

```
Exception ignored in: <generator object my_gen at 0x7f902073fe08>
RuntimeError: generator ignored GeneratorExit

Found 747 images belonging to 2 classes.

Exception ignored in: <generator object my_gen at 0x7f90207c3308>
RuntimeError: generator ignored GeneratorExit

Found 260 images belonging to 2 classes.

Exception ignored in: <generator object my_gen at 0x7f90207d10a0>
RuntimeError: generator ignored GeneratorExit
```

[0]:
```
train_features = np.reshape(train_features, (2890, 4 * 4 * 512))
validation_features = np.reshape(validation_features, (747, 4 * 4 * 512))
test_features = np.reshape(test_features, (260, 4 * 4 * 512))
```

0.1 Adding a densely-connected classifier on top of the convolutional base

[9]:
```
from keras import models          #Make 6 layers of VG16 model tarinable
from keras import layers
from keras import optimizers

model = models.Sequential()
# Fully connected layer
model.add(layers.Dense(output_dim = 128,
                     activation = 'relu', input_dim = 4 * 4 * 512))                ␣
 ↪    # Dense layer
model.add(layers.Dropout(0.25))                            # Dropout
model.add(layers.Dense(output_dim = 1,
                     activation = 'sigmoid'))            # Output

model.compile(optimizer=optimizers.RMSprop(lr=2e-5),
loss='binary_crossentropy',
metrics=['acc'])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:148: The name
tf.placeholder_with_default is deprecated. Please use
tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3733: calling dropout (from
tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed
in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 -
```

```
keep_prob`.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated.
Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3657: The name tf.log is
deprecated. Please use tf.math.log instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow_core/python/ops/nn_impl.py:183: where (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:8: UserWarning:
Update your `Dense` call to the Keras 2 API: `Dense(activation="relu",
input_dim=8192, units=128)`

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:11: UserWarning:
Update your `Dense` call to the Keras 2 API: `Dense(activation="sigmoid",
units=1)`
  # This is added back by InteractiveShellApp.init_path()
```

0.2 Summary of extended model

[10]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               1048704
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 129
=================================================================
Total params: 1,048,833
Trainable params: 1,048,833
Non-trainable params: 0
_________________________________________________________________
```

##Training

[11]:
```
history = model.fit(train_features, train_labels,
                    epochs=30,
                    batch_size=20,
                    validation_data=(validation_features, validation_labels))
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:1033: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:1020: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.

Train on 2890 samples, validate on 747 samples
Epoch 1/30
2890/2890 [==============================] - 1s 435us/step - loss: 0.5582 - acc:
0.7048 - val_loss: 0.4794 - val_acc: 0.7965
Epoch 2/30
2890/2890 [==============================] - 1s 326us/step - loss: 0.4037 - acc:
0.8246 - val_loss: 0.4300 - val_acc: 0.8233
Epoch 3/30
2890/2890 [==============================] - 1s 336us/step - loss: 0.3484 - acc:
0.8581 - val_loss: 0.4056 - val_acc: 0.8313
Epoch 4/30
2890/2890 [==============================] - 1s 340us/step - loss: 0.3157 - acc:
0.8754 - val_loss: 0.4269 - val_acc: 0.8206
Epoch 5/30
2890/2890 [==============================] - 1s 328us/step - loss: 0.2921 - acc:
0.8862 - val_loss: 0.3862 - val_acc: 0.8407
Epoch 6/30
2890/2890 [==============================] - 1s 331us/step - loss: 0.2677 - acc:
0.8986 - val_loss: 0.3899 - val_acc: 0.8394
Epoch 7/30
2890/2890 [==============================] - 1s 332us/step - loss: 0.2537 - acc:
0.9031 - val_loss: 0.3944 - val_acc: 0.8394
Epoch 8/30
2890/2890 [==============================] - 1s 337us/step - loss: 0.2404 - acc:
0.9076 - val_loss: 0.3686 - val_acc: 0.8367
Epoch 9/30
2890/2890 [==============================] - 1s 322us/step - loss: 0.2215 - acc:
0.9190 - val_loss: 0.3707 - val_acc: 0.8447
Epoch 10/30
2890/2890 [==============================] - 1s 310us/step - loss: 0.2155 - acc:
0.9263 - val_loss: 0.3761 - val_acc: 0.8487
Epoch 11/30
2890/2890 [==============================] - 1s 313us/step - loss: 0.2038 - acc:
0.9270 - val_loss: 0.3987 - val_acc: 0.8420
Epoch 12/30
2890/2890 [==============================] - 1s 315us/step - loss: 0.1909 - acc:
0.9329 - val_loss: 0.3642 - val_acc: 0.8474
```

```
Epoch 13/30
2890/2890 [==============================] - 1s 335us/step - loss: 0.1819 - acc:
0.9374 - val_loss: 0.3676 - val_acc: 0.8554
Epoch 14/30
2890/2890 [==============================] - 1s 324us/step - loss: 0.1730 - acc:
0.9408 - val_loss: 0.3645 - val_acc: 0.8514
Epoch 15/30
2890/2890 [==============================] - 1s 316us/step - loss: 0.1660 - acc:
0.9457 - val_loss: 0.3782 - val_acc: 0.8541
Epoch 16/30
2890/2890 [==============================] - 1s 319us/step - loss: 0.1588 - acc:
0.9450 - val_loss: 0.3709 - val_acc: 0.8541
Epoch 17/30
2890/2890 [==============================] - 1s 338us/step - loss: 0.1524 - acc:
0.9498 - val_loss: 0.3668 - val_acc: 0.8501
Epoch 18/30
2890/2890 [==============================] - 1s 320us/step - loss: 0.1459 - acc:
0.9516 - val_loss: 0.3764 - val_acc: 0.8501
Epoch 19/30
2890/2890 [==============================] - 1s 331us/step - loss: 0.1360 - acc:
0.9557 - val_loss: 0.3794 - val_acc: 0.8514
Epoch 20/30
2890/2890 [==============================] - 1s 320us/step - loss: 0.1380 - acc:
0.9599 - val_loss: 0.3799 - val_acc: 0.8487
Epoch 21/30
2890/2890 [==============================] - 1s 316us/step - loss: 0.1282 - acc:
0.9595 - val_loss: 0.3796 - val_acc: 0.8487
Epoch 22/30
2890/2890 [==============================] - 1s 335us/step - loss: 0.1220 - acc:
0.9647 - val_loss: 0.3823 - val_acc: 0.8487
Epoch 23/30
2890/2890 [==============================] - 1s 353us/step - loss: 0.1183 - acc:
0.9647 - val_loss: 0.3804 - val_acc: 0.8487
Epoch 24/30
2890/2890 [==============================] - 1s 339us/step - loss: 0.1168 - acc:
0.9633 - val_loss: 0.3841 - val_acc: 0.8487
Epoch 25/30
2890/2890 [==============================] - 1s 328us/step - loss: 0.1089 - acc:
0.9713 - val_loss: 0.4049 - val_acc: 0.8594
Epoch 26/30
2890/2890 [==============================] - 1s 335us/step - loss: 0.1049 - acc:
0.9696 - val_loss: 0.4116 - val_acc: 0.8541
Epoch 27/30
2890/2890 [==============================] - 1s 336us/step - loss: 0.1002 - acc:
0.9737 - val_loss: 0.4078 - val_acc: 0.8554
Epoch 28/30
2890/2890 [==============================] - 1s 339us/step - loss: 0.0960 - acc:
0.9734 - val_loss: 0.3955 - val_acc: 0.8501
```

```
Epoch 29/30
2890/2890 [==============================] - 1s 337us/step - loss: 0.0944 - acc:
0.9737 - val_loss: 0.3918 - val_acc: 0.8461
Epoch 30/30
2890/2890 [==============================] - 1s 342us/step - loss: 0.0869 - acc:
0.9799 - val_loss: 0.3947 - val_acc: 0.8447
```

[12]:
```
# list all data in history
print(history.history.keys())
```

```
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```

0.3 Save neural network structure

[13]:
```
from pathlib import Path
import os
transferModelFeatExtract_structure = model.to_json()
f = Path("gdrive/My Drive/Colab Notebooks/transferModelFeatExtract_structure.
 ↪json")
f.write_text(transferModelFeatExtract_structure)
```

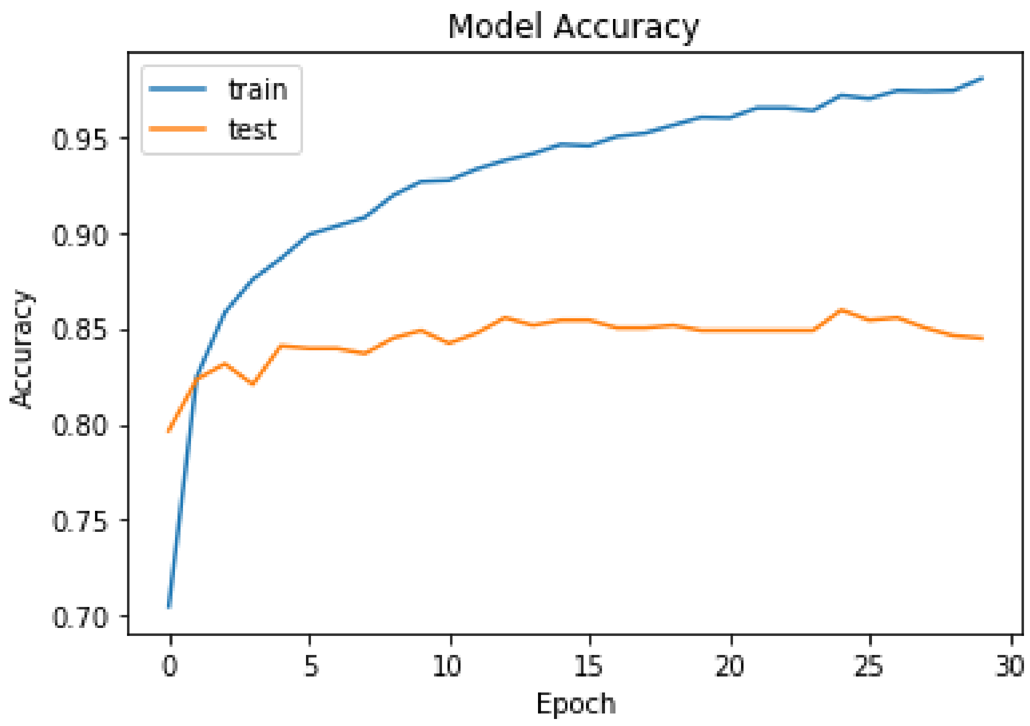
[13]:
```
1285
```

0.4 Save neural network's trained weights

[0]:
```
# Save neural network's trained weights
model.save_weights("gdrive/My Drive/Colab Notebooks/
 ↪transferModelFeatExtract_weights.h5")
model.save('gdrive/My Drive/Colab Notebooks/transferModelFeatExtract_weights.
 ↪h5')
```

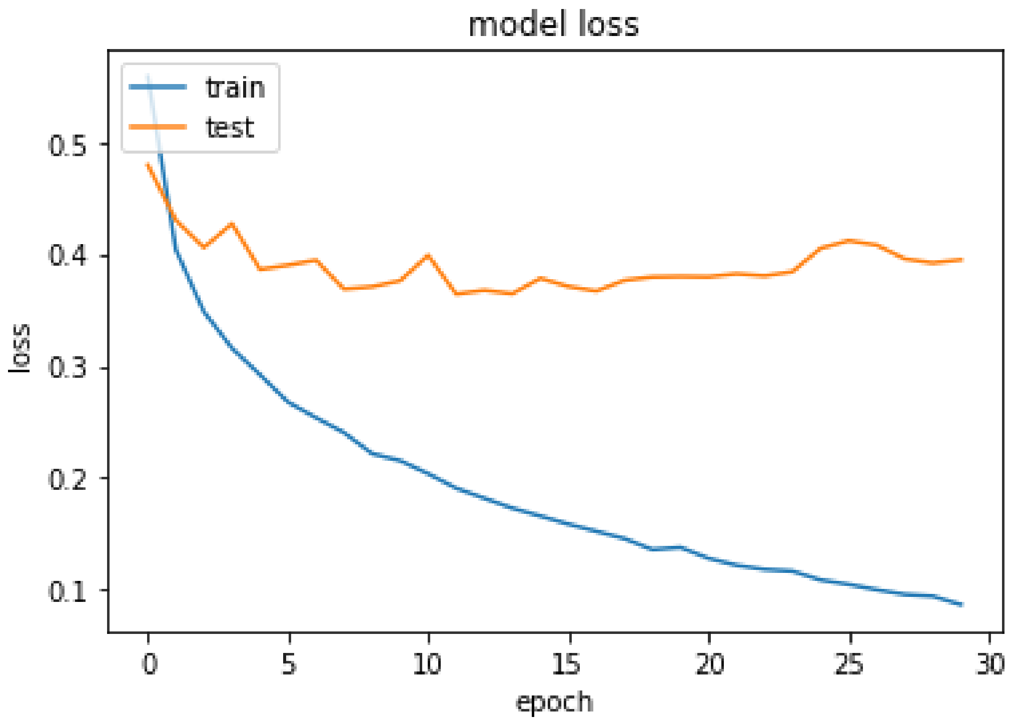
0.5 Plot training and validation accuracy per epoch

[15]:
```
import matplotlib.pyplot as plt
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



0.6 Plot training and validation loss per epoch

[16]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```


model loss
train
test
loss
epoch
0.1
0.2
0.3
0.4
0.5
0
5
10
15
20
25
30

cars_transfer_learning_feat_extract_with_data_aug

October 19, 2019

#CNN model of vehicle recognition - Feature extraction with data augmentation

[1]:
```
from google.colab import drive, files
drive.mount('/content/gdrive')
gdrive_path = 'gdrive/My\ Drive/Colab\ Notebooks/'
!rsync -ah --progress\
    {gdrive_path}cars_dl/*.zip\
    {gdrive_path}model_weights_final*.h5\
    {gdrive_path}model_struct_final*.json\
    '/content'
!unzip -qo '*.zip'
!rm *.zip
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id
=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redire
ct_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2Fwww.googl
eapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive%2
0https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2
Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code

Enter your authorization code:
ůůůůůůůůůů
Mounted at /content/gdrive
sending incremental file list
rsync: link_stat "/content/gdrive/My Drive/Colab
Notebooks/model_weights_final*.h5" failed: No such file or directory (2)
rsync: link_stat "/content/gdrive/My Drive/Colab
Notebooks/model_structure_final*.json" failed: No such file or directory (2)
cars-20191002T050118Z-001.zip
        490.68M 100%   54.52MB/s    0:00:08 (xfr#1, to-chk=0/1)
rsync error: some files/attrs were not transferred (see previous errors) (code
23) at main.c(1196) [sender=3.1.2]
```

[2]:
```
# Load Keras' VGG16 model that was pre-trained against the ImageNet database
from keras.applications import VGG16
conv_base = VGG16(weights='imagenet',
include_top=False,
input_shape=(128, 128, 3))
```

```
Using TensorFlow backend.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:66: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4432: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4267: The name tf.nn.max_pool is
deprecated. Please use tf.nn.max_pool2d instead.

Downloading data from https://github.com/fchollet/deep-learning-
models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 1s 0us/step
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:190: The name
tf.get_default_session is deprecated. Please use
tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:197: The name tf.ConfigProto is
deprecated. Please use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:203: The name tf.Session is
deprecated. Please use tf.compat.v1.Session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:207: The name tf.global_variables
is deprecated. Please use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:216: The name
tf.is_variable_initialized is deprecated. Please use
tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:223: The name
tf.variables_initializer is deprecated. Please use
tf.compat.v1.variables_initializer instead.
```

##Add dense classifier on top of convolutional base

[0]:
```
from keras import models
from keras import layers
model = models.Sequential()
model.add(conv_base)
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

[4]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               2097408
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257
=================================================================
Total params: 16,812,353
Trainable params: 16,812,353
Non-trainable params: 0
_________________________________________________________________
```

0.1 Freezing the convolutional base

[5]:
```
print('This is the number of trainable weights before freezing the conv base:',␣
 ↪len(model.trainable_weights))
```

```
This is the number of trainable weights before freezing the conv base: 30
```

[6]:
```
conv_base.trainable = False
print('This is the number of trainable weights after freezing the conv base:',␣
 ↪len(model.trainable_weights))
```

```
This is the number of trainable weights after freezing the conv base: 4
```

[0]:
```
import numpy as np
import matplotlib.pyplot as plt
```

```python
#Method to print several images in a single row
def plots(img, figsize=(12,6), rows = 1, titles = 1):
    if type(img[0]) is np.ndarray:
        img = np.array(img).astype(np.float_)
        if (img.shape[-1] != 3):
            img = img.transpose((0, 2, 3, 1))
    f =plt.figure(figsize = figsize)
    cols = 7//rows if (len(img) % 2 == 0) else len(img)//rows + 1
    for i in range(cols):
        sp = f.add_subplot(rows, cols, i+1)
        sp.axis('Off')
        if titles is not None:
            sp.set_title(titles[i], fontsize = 10)
        plt.imshow(img[i], interpolation = None if np.interp else 'none')
```

[8]:
```python
import os

trainingCarImages = os.listdir('cars/training_set/car/')
print ("Number of Training car images - ",str(len(trainingCarImages)))

trainingCrashImages = os.listdir('cars/training_set/crash/')
print ("Number of Training crash images - ",str(len(trainingCrashImages)))

validCarImages = os.listdir('cars/val_set/car/')
print ("Number of Validation car images - ",str(len(validCarImages)))

validCrashImages = os.listdir('cars/val_set/crash/')
print ("Number of Validation crash images - ",str(len(validCrashImages)))

testCarImages = os.listdir('cars/test_set/car/')
print ("Number of Test car images - ",str(len(testCarImages)))

testCrashImages = os.listdir('cars/test_set/crash/')
print ("Number of Test crash images - ",str(len(testCrashImages)))

train_data_dir = 'cars/training_set/'
validation_data_dir = 'cars/val_set/'
test_data_dir = 'cars/test_set/'
nb_train_samples = trainingCarImages + trainingCrashImages
nb_validation_samples = validCarImages + validCrashImages
nb_test_samples = testCarImages + testCrashImages
```

```
Number of Training car images -  1445
Number of Training crash images -  1445
Number of Validation car images -  375
Number of Validation crash images -  372
Number of Test car images -  129
Number of Test crash images -  131
```

0.2 Training the model end-to-end with a frozen convolutional base

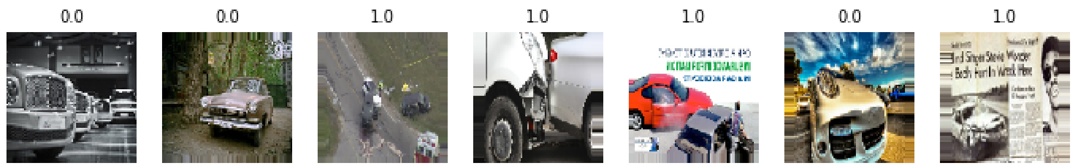
[0]:
```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
```

[10]:
```
training_set = train_datagen.flow_from_directory(train_data_dir,
                                                 target_size = (128, 128),
                                                 batch_size = 32,
                                                 class_mode = 'binary',
                                                 shuffle=True,
                                                 seed=42)

test_datagen = ImageDataGenerator(rescale = 1./255)

val_set = test_datagen.flow_from_directory(validation_data_dir,
                                            target_size = (128, 128),
                                            batch_size = 32,
                                            class_mode = 'binary',
                                            shuffle=True,
                                            seed=42)

test_set = test_datagen.flow_from_directory(test_data_dir,
                                            target_size = (128, 128),
                                            batch_size = 32,
                                            class_mode = 'binary',
                                            shuffle=True,
                                            seed=42)
```

```
Found 2890 images belonging to 2 classes.
Found 747 images belonging to 2 classes.
Found 260 images belonging to 2 classes.
```

0.3 Print normalized and re-sized images from training set

[11]:
```
imgs, labels = next(training_set)
plots(imgs, titles = labels)
```



0.4 Print normalized and re-sized images from validation set

[12]:
```
imgs, labels = next(val_set)
plots(imgs, titles = labels)
```

0.5 Compile model

[13]:
```
from keras import optimizers
model.compile(optimizer = optimizers.RMSprop(lr=2e-5), loss =␣
 ↪'binary_crossentropy', metrics = ['accuracy'])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated.
Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3657: The name tf.log is
deprecated. Please use tf.math.log instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow_core/python/ops/nn_impl.py:183: where (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

[0]:
```
#Method to identify and remove corrupted images
def my_gen(gen):
    while True:
        try:
            imgs, labels = next(gen)
            yield imgs, labels
        except:
            pass
```

0.6 Train model

[15]:
```
history = model.fit_generator(my_gen(training_set),
                              steps_per_epoch = 78,
                              epochs = 30,
                              validation_data = val_set,
                              validation_steps = 50)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:1033: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:1020: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.

Epoch 1/30

/usr/local/lib/python3.6/dist-packages/PIL/Image.py:914: UserWarning: Palette
images with Transparency   expressed in bytes should be converted to RGBA images
  'to RGBA images')

78/78 [==============================] - 47s 605ms/step - loss: 0.5565 - acc:
0.7336 - val_loss: 0.4780 - val_acc: 0.8049
Epoch 2/30
78/78 [==============================] - 40s 507ms/step - loss: 0.4385 - acc:
0.8167 - val_loss: 0.4290 - val_acc: 0.8216
Epoch 3/30
78/78 [==============================] - 39s 494ms/step - loss: 0.3867 - acc:
0.8420 - val_loss: 0.4059 - val_acc: 0.8357
Epoch 4/30
78/78 [==============================] - 39s 502ms/step - loss: 0.3640 - acc:
0.8589 - val_loss: 0.4020 - val_acc: 0.8363
Epoch 5/30
78/78 [==============================] - 39s 501ms/step - loss: 0.3360 - acc:
0.8661 - val_loss: 0.3789 - val_acc: 0.8453
Epoch 6/30
78/78 [==============================] - 38s 486ms/step - loss: 0.3323 - acc:
0.8680 - val_loss: 0.4076 - val_acc: 0.8331
Epoch 7/30
78/78 [==============================] - 39s 500ms/step - loss: 0.3177 - acc:
0.8704 - val_loss: 0.3859 - val_acc: 0.8492
Epoch 8/30
78/78 [==============================] - 39s 500ms/step - loss: 0.2967 - acc:
0.8789 - val_loss: 0.4942 - val_acc: 0.7908
Epoch 9/30
78/78 [==============================] - 39s 497ms/step - loss: 0.2980 - acc:
0.8752 - val_loss: 0.4240 - val_acc: 0.8216
```

```
Epoch 10/30
78/78 [==============================] - 38s 494ms/step - loss: 0.2759 - acc:
0.8918 - val_loss: 0.3678 - val_acc: 0.8569
Epoch 11/30
78/78 [==============================] - 40s 514ms/step - loss: 0.2818 - acc:
0.8853 - val_loss: 0.3921 - val_acc: 0.8466
Epoch 12/30
78/78 [==============================] - 36s 468ms/step - loss: 0.2840 - acc:
0.8854 - val_loss: 0.4177 - val_acc: 0.8328
Epoch 13/30
78/78 [==============================] - 40s 507ms/step - loss: 0.2679 - acc:
0.8882 - val_loss: 0.3906 - val_acc: 0.8440
Epoch 14/30
78/78 [==============================] - 39s 500ms/step - loss: 0.2537 - acc:
0.9002 - val_loss: 0.4087 - val_acc: 0.8421
Epoch 15/30
78/78 [==============================] - 39s 500ms/step - loss: 0.2580 - acc:
0.8974 - val_loss: 0.3733 - val_acc: 0.8549
Epoch 16/30
78/78 [==============================] - 38s 484ms/step - loss: 0.2453 - acc:
0.9090 - val_loss: 0.4092 - val_acc: 0.8434
Epoch 17/30
78/78 [==============================] - 40s 513ms/step - loss: 0.2541 - acc:
0.9014 - val_loss: 0.3655 - val_acc: 0.8601
Epoch 18/30
78/78 [==============================] - 40s 510ms/step - loss: 0.2324 - acc:
0.9134 - val_loss: 0.4714 - val_acc: 0.8094
Epoch 19/30
78/78 [==============================] - 38s 491ms/step - loss: 0.2412 - acc:
0.9028 - val_loss: 0.3756 - val_acc: 0.8530
Epoch 20/30
78/78 [==============================] - 39s 497ms/step - loss: 0.2402 - acc:
0.9061 - val_loss: 0.4006 - val_acc: 0.8395
Epoch 21/30
78/78 [==============================] - 39s 498ms/step - loss: 0.2375 - acc:
0.9038 - val_loss: 0.3910 - val_acc: 0.8479
Epoch 22/30
78/78 [==============================] - 39s 496ms/step - loss: 0.2223 - acc:
0.9142 - val_loss: 0.3988 - val_acc: 0.8472
Epoch 23/30
78/78 [==============================] - 36s 467ms/step - loss: 0.2265 - acc:
0.9147 - val_loss: 0.3870 - val_acc: 0.8569
Epoch 24/30
78/78 [==============================] - 40s 515ms/step - loss: 0.2219 - acc:
0.9171 - val_loss: 0.4376 - val_acc: 0.8302
Epoch 25/30
78/78 [==============================] - 38s 484ms/step - loss: 0.2233 - acc:
0.9130 - val_loss: 0.4488 - val_acc: 0.8254
```

```
Epoch 26/30
78/78 [==============================] - 39s 495ms/step - loss: 0.2166 - acc:
0.9159 - val_loss: 0.4211 - val_acc: 0.8344
Epoch 27/30
78/78 [==============================] - 39s 500ms/step - loss: 0.2084 - acc:
0.9195 - val_loss: 0.4464 - val_acc: 0.8306
Epoch 28/30
78/78 [==============================] - 38s 488ms/step - loss: 0.2024 - acc:
0.9283 - val_loss: 0.3917 - val_acc: 0.8543
Epoch 29/30
78/78 [==============================] - 39s 504ms/step - loss: 0.2142 - acc:
0.9131 - val_loss: 0.3768 - val_acc: 0.8575
Epoch 30/30
78/78 [==============================] - 39s 501ms/step - loss: 0.1974 - acc:
0.9313 - val_loss: 0.3978 - val_acc: 0.8633

Exception ignored in: <generator object my_gen at 0x7f101ad8ad00>
RuntimeError: generator ignored GeneratorExit
```

[16]:
```
# list all data in history
print(history.history.keys())
```

```
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```

0.7 Save neural network structure

[17]:
```
from pathlib import Path
import os
transferModelFeatExtractDataAug_structure = model.to_json()
f = Path("transferModelFeatExtractDataAug_structure.json")
f.write_text(transferModelFeatExtractDataAug_structure)
```

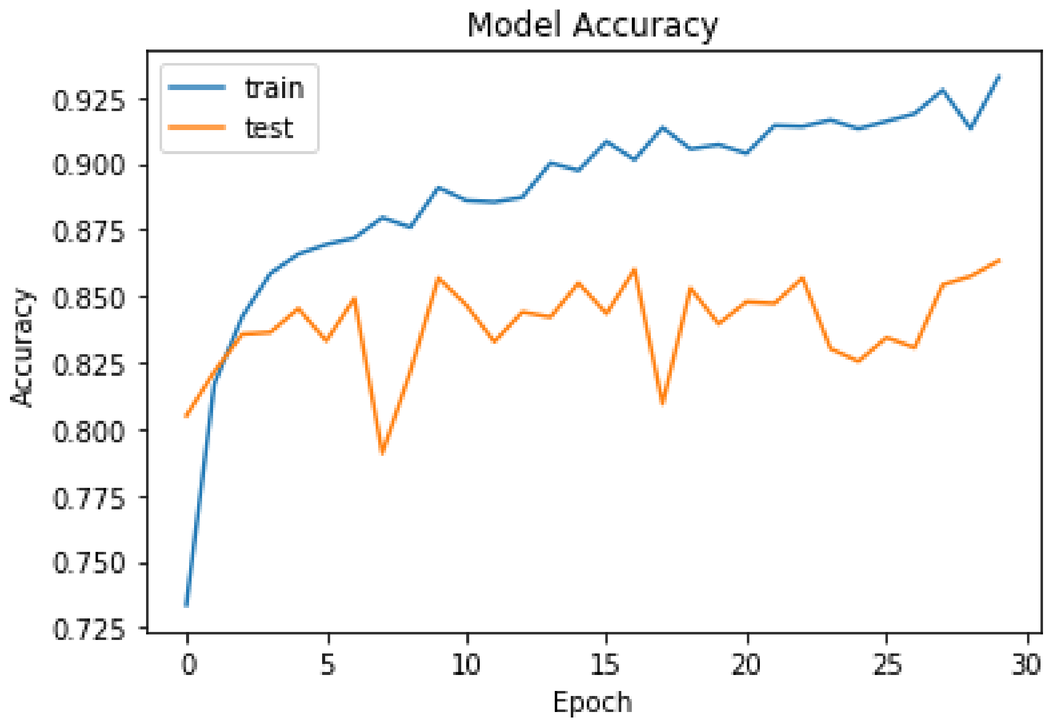
[17]:
```
11815
```

0.8 Save neural network's trained weights

[0]:
```
# Save neural network's trained weights
model.save_weights("transferModelFeatExtractDataAug_weights.h5")
model.save('transferModelFeatExtractDataAug_weights.h5')
```

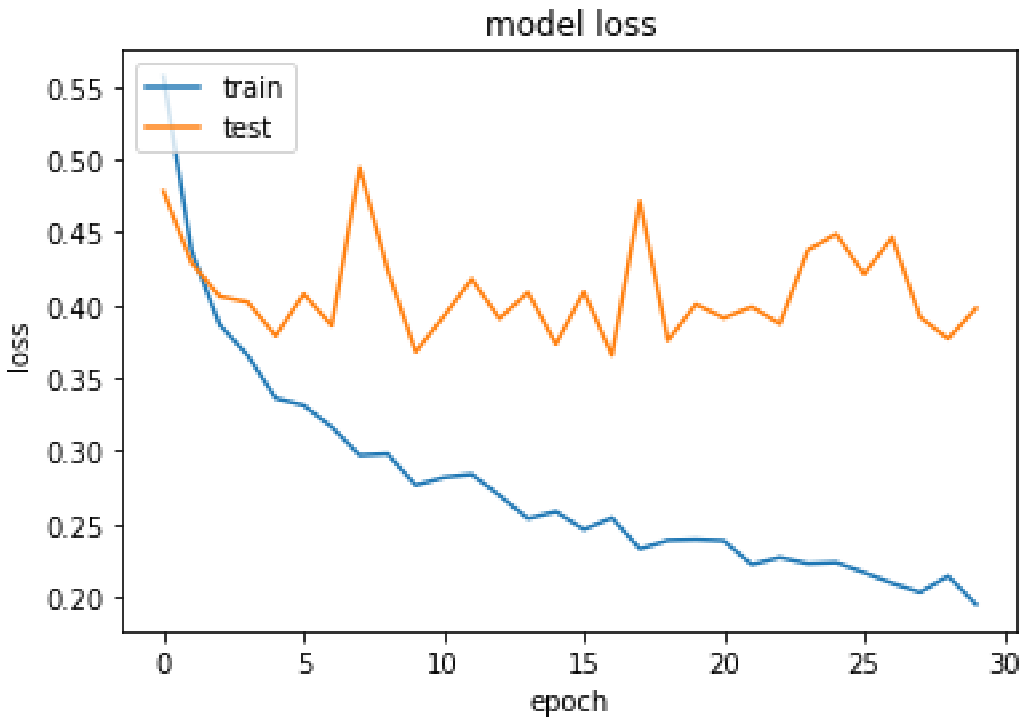
0.9 Plot training and validation accuracy per epoch

[19]:
```
import matplotlib.pyplot as plt
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



0.10 Plot training and validation loss per epoch

[20]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```


model loss
train
test
loss
epoch
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0
5
10
15
20
25
30

cars_transfer_learning_fine_tuning

October 19, 2019

#CNN model of vehicle recognition - Fine tuning

[1]:
```
from google.colab import drive, files
drive.mount('/content/gdrive')
gdrive_path = 'gdrive/My\ Drive/Colab\ Notebooks/'
!rsync -ah --progress\
    {gdrive_path}cars_dl/*.zip\
    '/content'
!unzip -qo '*.zip'
!rm *.zip
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id
=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redire
ct_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2Fwww.googl
eapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive%2
0https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2
Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code

Enter your authorization code:
ůůůůůůůůůů
Mounted at /content/gdrive
sending incremental file list
cars-20191002T050118Z-001.zip
        490.68M 100%   82.80MB/s    0:00:05 (xfr#1, to-chk=0/1)
```

[0]:
```
epochs = 50 #@param {type:"number"}
validation_steps = 32 #@param {type:"number"}
img_height = 128 #@param {type:"integer"}
img_width = 128 #@param {type:"integer"}
batch_size = 32 #@param {type:"number"}
```

##Example of using a pre-trained model (VGG16) as a classifier

[3]:
```
# Load Keras' VGG16 model that was pre-trained against the ImageNet database
from keras.applications import VGG16
conv_base = VGG16(weights='imagenet',
include_top=False,
input_shape=(128, 128, 3))
```

```
Using TensorFlow backend.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:66: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4432: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4267: The name tf.nn.max_pool is
deprecated. Please use tf.nn.max_pool2d instead.

Downloading data from https://github.com/fchollet/deep-learning-
models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 1s 0us/step
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:190: The name
tf.get_default_session is deprecated. Please use
tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:197: The name tf.ConfigProto is
deprecated. Please use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:203: The name tf.Session is
deprecated. Please use tf.compat.v1.Session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:207: The name tf.global_variables
is deprecated. Please use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:216: The name
tf.is_variable_initialized is deprecated. Please use
tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:223: The name
tf.variables_initializer is deprecated. Please use
tf.compat.v1.variables_initializer instead.
```

[4]:
```
# Summary of Keras' VGG16 model that was pre-trained against the ImageNet␣
↪database
conv_base.summary()
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 128, 128, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 128, 128, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 128, 128, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 64, 64, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 64, 64, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 64, 64, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 32, 32, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 32, 32, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 32, 32, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 32, 32, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 16, 16, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 16, 16, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 8, 8, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 4, 4, 512)         0
=================================================================
Total params: 14,714,688
```

```
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

##Freeze all layers up to a specific one

[0]:
```
conv_base.trainable = True

set_trainable = False
for layer in conv_base.layers:
  if layer.name == 'block4_conv2':
    set_trainable = True
  if set_trainable:
    layer.trainable = True
  else:
    layer.trainable = False
```

[0]:
```
from keras import models
from keras import layers
model = models.Sequential()
model.add(conv_base)
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

[7]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               2097408
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257
=================================================================
Total params: 16,812,353
Trainable params: 13,896,705
Non-trainable params: 2,915,648
_________________________________________________________________
```

[0]:
```
import numpy as np
import matplotlib.pyplot as plt

#Method to print several images in a single row
def plots(img, figsize=(12,6), rows = 1, titles = 1):
    if type(img[0]) is np.ndarray:
        img = np.array(img).astype(np.float_)
        if (img.shape[-1] != 3):
            img = img.transpose((0, 2, 3, 1))
    f =plt.figure(figsize = figsize)
    cols = 7//rows if (len(img) % 2 == 0) else len(img)//rows + 1
    for i in range(cols):
        sp = f.add_subplot(rows, cols, i+1)
        sp.axis('Off')
        if titles is not None:
            sp.set_title(titles[i], fontsize = 10)
        plt.imshow(img[i], interpolation = None if np.interp else 'none')
```

[9]:
```
import os

trainingCarImages = os.listdir('cars/training_set/car/')
print ("Number of Training car images - ",str(len(trainingCarImages)))

trainingCrashImages = os.listdir('cars/training_set/crash/')
print ("Number of Training crash images - ",str(len(trainingCrashImages)))

validCarImages = os.listdir('cars/val_set/car/')
print ("Number of Validation car images - ",str(len(validCarImages)))

validCrashImages = os.listdir('cars/val_set/crash/')
print ("Number of Validation crash images - ",str(len(validCrashImages)))

testCarImages = os.listdir('cars/test_set/car/')
print ("Number of Test car images - ",str(len(testCarImages)))

testCrashImages = os.listdir('cars/test_set/crash/')
print ("Number of Test crash images - ",str(len(testCrashImages)))

train_data_dir = 'cars/training_set/'
validation_data_dir = 'cars/val_set/'
test_data_dir = 'cars/test_set/'
nb_train_samples = trainingCarImages + trainingCrashImages
nb_validation_samples = validCarImages + validCrashImages
nb_test_samples = testCarImages + testCrashImages
```

```
Number of Training car images -  1445
Number of Training crash images -  1445
Number of Validation car images -  375
Number of Validation crash images -  372
Number of Test car images -  129
Number of Test crash images -  131
```

0.1 Training the model end-to-end with a frozen convolutional base

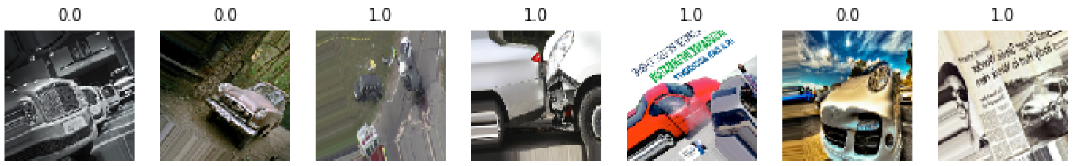
[0]:
```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=40,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
```

[11]:
```
training_set = train_datagen.flow_from_directory(train_data_dir,
                                                 target_size = (128, 128),
                                                 batch_size = 32,
                                                 class_mode = 'binary',
                                                 shuffle=True,
                                                 seed=42)

test_datagen = ImageDataGenerator(rescale = 1./255)

val_set = test_datagen.flow_from_directory(validation_data_dir,
                                           target_size = (128, 128),
                                           batch_size = 32,
                                           class_mode = 'binary',
                                           shuffle=True,
                                           seed=42)

test_set = test_datagen.flow_from_directory(test_data_dir,
                                            target_size = (128, 128),
                                            batch_size = 32,
                                            class_mode = 'binary',
                                            shuffle=True,
                                            seed=42)
```

```
Found 2890 images belonging to 2 classes.
Found 747 images belonging to 2 classes.
Found 260 images belonging to 2 classes.
```

0.2 Print normalized and re-sized images from training set

[12]:
```
imgs, labels = next(training_set)
plots(imgs, titles = labels)
```



0.3 Print normalized and re-sized images from validation set

[13]:
```
imgs, labels = next(val_set)
plots(imgs, titles = labels)
```

0.4 Compile model

[14]:
```
from keras import optimizers
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics =␣
 ↪['accuracy'])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated.
Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3657: The name tf.log is
deprecated. Please use tf.math.log instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow_core/python/ops/nn_impl.py:183: where (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

[0]:
```
#Method to identify and remove corrupted images
def my_gen(gen):
    while True:
        try:
            imgs, labels = next(gen)
            yield imgs, labels
        except:
            pass
```

##Fine-tuning the model

[16]:
```
model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(lr=1e-5),
              metrics=['acc'])

history = model.fit_generator(
    my_gen(training_set),
    steps_per_epoch=100,
    epochs=50,
    validation_data=val_set,
    validation_steps=50)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:1033: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:1020: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.

Epoch 1/50

/usr/local/lib/python3.6/dist-packages/PIL/Image.py:914: UserWarning: Palette
images with Transparency   expressed in bytes should be converted to RGBA images
  'to RGBA images')

100/100 [==============================] - 59s 592ms/step - loss: 0.5148 - acc:
0.7426 - val_loss: 0.5168 - val_acc: 0.7805
Epoch 2/50
100/100 [==============================] - 52s 524ms/step - loss: 0.3964 - acc:
0.8271 - val_loss: 0.4756 - val_acc: 0.8081
Epoch 3/50
100/100 [==============================] - 53s 527ms/step - loss: 0.3535 - acc:
0.8487 - val_loss: 0.4947 - val_acc: 0.8055
Epoch 4/50
100/100 [==============================] - 52s 516ms/step - loss: 0.3343 - acc:
0.8534 - val_loss: 0.5627 - val_acc: 0.7792
Epoch 5/50
100/100 [==============================] - 51s 514ms/step - loss: 0.3015 - acc:
```

```
0.8779 - val_loss: 0.3834 - val_acc: 0.8614
Epoch 6/50
100/100 [==============================] - 51s 509ms/step - loss: 0.2864 - acc:
0.8809 - val_loss: 0.4051 - val_acc: 0.8517
Epoch 7/50
100/100 [==============================] - 52s 516ms/step - loss: 0.2652 - acc:
0.8962 - val_loss: 0.4359 - val_acc: 0.8421
Epoch 8/50
100/100 [==============================] - 52s 517ms/step - loss: 0.2463 - acc:
0.9062 - val_loss: 0.3701 - val_acc: 0.8678
Epoch 9/50
100/100 [==============================] - 51s 512ms/step - loss: 0.2434 - acc:
0.9018 - val_loss: 0.6857 - val_acc: 0.7734
Epoch 10/50
100/100 [==============================] - 54s 535ms/step - loss: 0.2191 - acc:
0.9144 - val_loss: 0.5260 - val_acc: 0.8402
Epoch 11/50
100/100 [==============================] - 53s 526ms/step - loss: 0.2160 - acc:
0.9121 - val_loss: 0.5570 - val_acc: 0.8235
Epoch 12/50
100/100 [==============================] - 52s 524ms/step - loss: 0.2098 - acc:
0.9205 - val_loss: 0.3625 - val_acc: 0.8842
Epoch 13/50
100/100 [==============================] - 52s 517ms/step - loss: 0.2020 - acc:
0.9234 - val_loss: 0.4513 - val_acc: 0.8575
Epoch 14/50
100/100 [==============================] - 52s 518ms/step - loss: 0.1862 - acc:
0.9239 - val_loss: 0.5983 - val_acc: 0.8184
Epoch 15/50
100/100 [==============================] - 50s 502ms/step - loss: 0.1869 - acc:
0.9256 - val_loss: 0.4147 - val_acc: 0.8646
Epoch 16/50
100/100 [==============================] - 53s 526ms/step - loss: 0.1923 - acc:
0.9265 - val_loss: 0.5003 - val_acc: 0.8363
Epoch 17/50
100/100 [==============================] - 50s 503ms/step - loss: 0.1726 - acc:
0.9309 - val_loss: 0.6170 - val_acc: 0.8331
Epoch 18/50
100/100 [==============================] - 49s 492ms/step - loss: 0.1499 - acc:
0.9415 - val_loss: 1.2837 - val_acc: 0.7368
Epoch 19/50
100/100 [==============================] - 52s 515ms/step - loss: 0.1427 - acc:
0.9456 - val_loss: 0.7777 - val_acc: 0.7972
Epoch 20/50
100/100 [==============================] - 52s 521ms/step - loss: 0.1547 - acc:
0.9434 - val_loss: 0.5618 - val_acc: 0.8331
Epoch 21/50
100/100 [==============================] - 52s 516ms/step - loss: 0.1381 - acc:
```

```
0.9497 - val_loss: 0.4380 - val_acc: 0.8703
Epoch 22/50
100/100 [==============================] - 51s 509ms/step - loss: 0.1266 - acc:
0.9516 - val_loss: 0.6637 - val_acc: 0.8402
Epoch 23/50
100/100 [==============================] - 54s 536ms/step - loss: 0.1337 - acc:
0.9428 - val_loss: 0.6278 - val_acc: 0.8325
Epoch 24/50
100/100 [==============================] - 52s 519ms/step - loss: 0.1283 - acc:
0.9497 - val_loss: 0.5907 - val_acc: 0.8471
Epoch 25/50
100/100 [==============================] - 51s 514ms/step - loss: 0.1199 - acc:
0.9528 - val_loss: 0.5215 - val_acc: 0.8716
Epoch 26/50
100/100 [==============================] - 51s 509ms/step - loss: 0.1038 - acc:
0.9594 - val_loss: 0.8867 - val_acc: 0.8004
Epoch 27/50
100/100 [==============================] - 51s 506ms/step - loss: 0.1012 - acc:
0.9603 - val_loss: 0.6917 - val_acc: 0.8479
Epoch 28/50
100/100 [==============================] - 53s 531ms/step - loss: 0.0989 - acc:
0.9603 - val_loss: 0.6396 - val_acc: 0.8582
Epoch 29/50
100/100 [==============================] - 51s 507ms/step - loss: 0.0916 - acc:
0.9650 - val_loss: 0.9350 - val_acc: 0.8177
Epoch 30/50
100/100 [==============================] - 52s 522ms/step - loss: 0.0941 - acc:
0.9653 - val_loss: 0.5749 - val_acc: 0.8800
Epoch 31/50
100/100 [==============================] - 52s 520ms/step - loss: 0.0883 - acc:
0.9669 - val_loss: 0.9241 - val_acc: 0.8196
Epoch 32/50
100/100 [==============================] - 52s 517ms/step - loss: 0.0832 - acc:
0.9672 - val_loss: 0.8796 - val_acc: 0.8318
Epoch 33/50
100/100 [==============================] - 53s 535ms/step - loss: 0.0824 - acc:
0.9700 - val_loss: 1.0892 - val_acc: 0.7792
Epoch 34/50
100/100 [==============================] - 50s 503ms/step - loss: 0.0690 - acc:
0.9759 - val_loss: 1.0543 - val_acc: 0.8261
Epoch 35/50
100/100 [==============================] - 52s 519ms/step - loss: 0.0791 - acc:
0.9712 - val_loss: 0.7413 - val_acc: 0.8588
Epoch 36/50
100/100 [==============================] - 52s 524ms/step - loss: 0.0768 - acc:
0.9700 - val_loss: 0.6897 - val_acc: 0.8608
Epoch 37/50
100/100 [==============================] - 51s 509ms/step - loss: 0.0620 - acc:
```

```
0.9781 - val_loss: 1.0443 - val_acc: 0.8318
Epoch 38/50
100/100 [==============================] - 52s 518ms/step - loss: 0.0659 - acc:
0.9775 - val_loss: 1.0645 - val_acc: 0.8241
Epoch 39/50
100/100 [==============================] - 50s 501ms/step - loss: 0.0688 - acc:
0.9766 - val_loss: 0.6178 - val_acc: 0.8652
Epoch 40/50
100/100 [==============================] - 51s 510ms/step - loss: 0.0626 - acc:
0.9749 - val_loss: 0.9118 - val_acc: 0.8447
Epoch 41/50
100/100 [==============================] - 52s 525ms/step - loss: 0.0583 - acc:
0.9787 - val_loss: 0.9474 - val_acc: 0.8254
Epoch 42/50
100/100 [==============================] - 52s 516ms/step - loss: 0.0602 - acc:
0.9781 - val_loss: 0.9252 - val_acc: 0.8338
Epoch 43/50
100/100 [==============================] - 49s 492ms/step - loss: 0.0535 - acc:
0.9806 - val_loss: 0.7929 - val_acc: 0.8646
Epoch 44/50
100/100 [==============================] - 52s 523ms/step - loss: 0.0465 - acc:
0.9831 - val_loss: 0.9719 - val_acc: 0.8331
Epoch 45/50
100/100 [==============================] - 51s 514ms/step - loss: 0.0515 - acc:
0.9828 - val_loss: 0.8152 - val_acc: 0.8697
Epoch 46/50
100/100 [==============================] - 52s 521ms/step - loss: 0.0517 - acc:
0.9794 - val_loss: 1.1029 - val_acc: 0.8434
Epoch 47/50
100/100 [==============================] - 53s 529ms/step - loss: 0.0557 - acc:
0.9797 - val_loss: 1.0190 - val_acc: 0.8383
Epoch 48/50
100/100 [==============================] - 52s 515ms/step - loss: 0.0548 - acc:
0.9803 - val_loss: 1.2516 - val_acc: 0.8256
Epoch 49/50
100/100 [==============================] - 51s 510ms/step - loss: 0.0448 - acc:
0.9812 - val_loss: 0.9756 - val_acc: 0.8460
Epoch 50/50
100/100 [==============================] - 52s 517ms/step - loss: 0.0539 - acc:
0.9793 - val_loss: 1.0245 - val_acc: 0.8370
```

[17]:
```python
# list all data in history
print(history.history.keys())
```

```
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```

0.5 Save neural network structure

[18]:
```python
from pathlib import Path
import os
transferModelFineTune_structure = model.to_json()
f = Path("gdrive/My Drive/Colab Notebooks/transferModelFineTune_structure.json")
f.write_text(transferModelFineTune_structure)
```

[18]:
```
11826
```

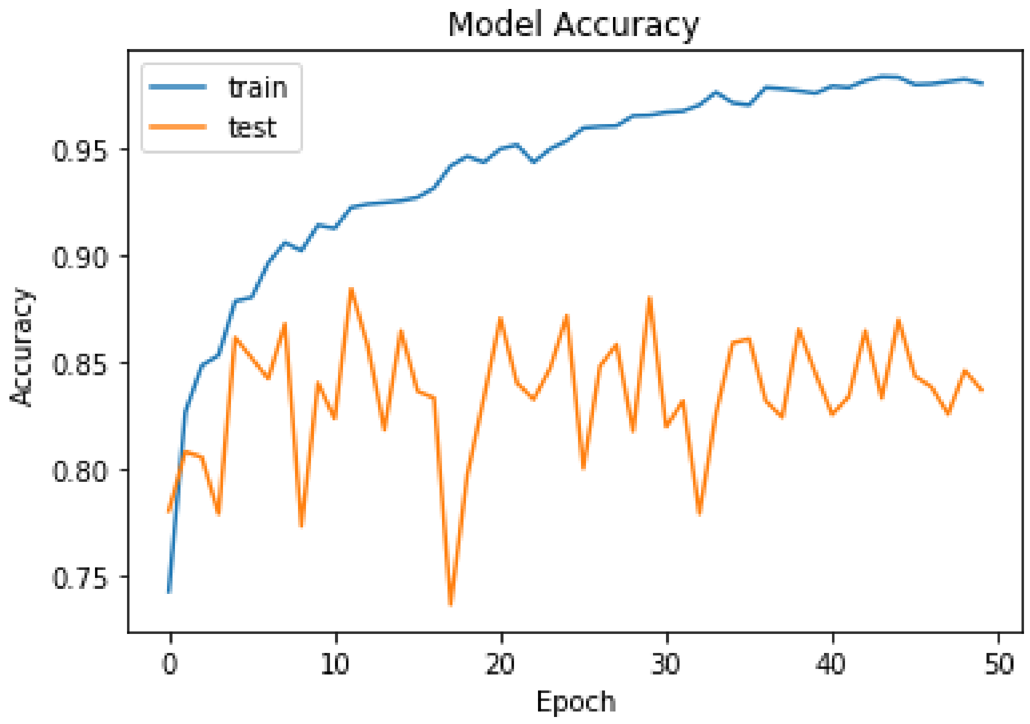
0.6 Save neural network's trained weights

[0]:
```python
# Save neural network's trained weights
model.save_weights("gdrive/My Drive/Colab Notebooks/
 ↪transferModelFineTune_weights.h5")
model.save('gdrive/My Drive/Colab Notebooks/transferModelFineTune_weights.h5')
```

0.7 Plot training and validation accuracy per epoch

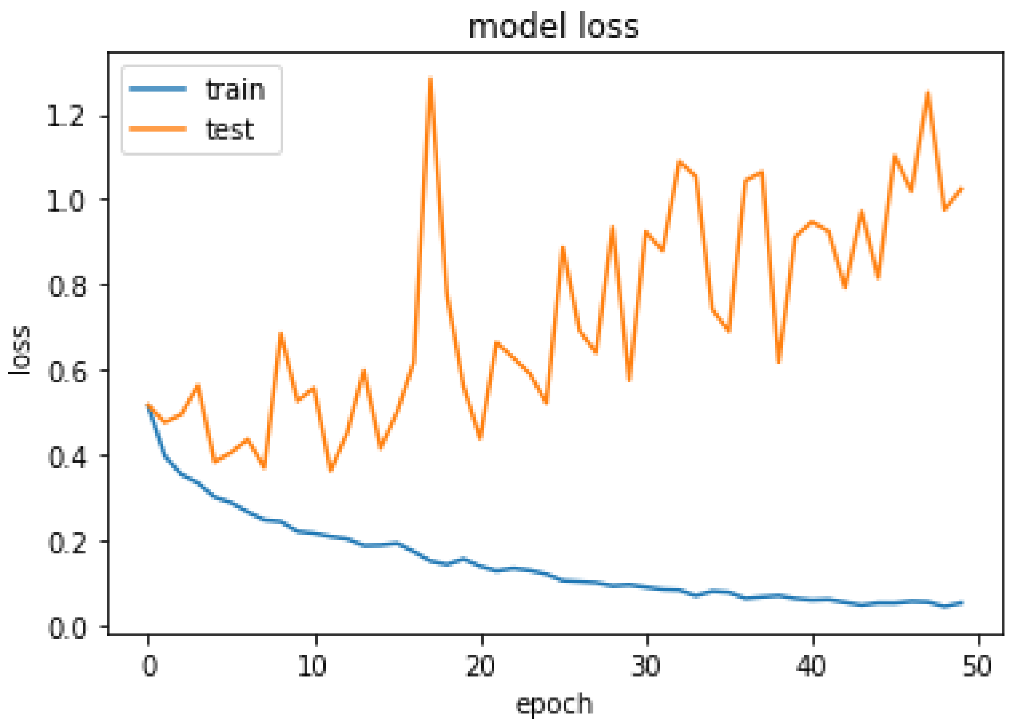
[20]:
```python
import matplotlib.pyplot as plt
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

```
Exception ignored in: <generator object my_gen at 0x7f1570237678>
RuntimeError: generator ignored GeneratorExit
```



0.8 Plot training and validation loss per epoch

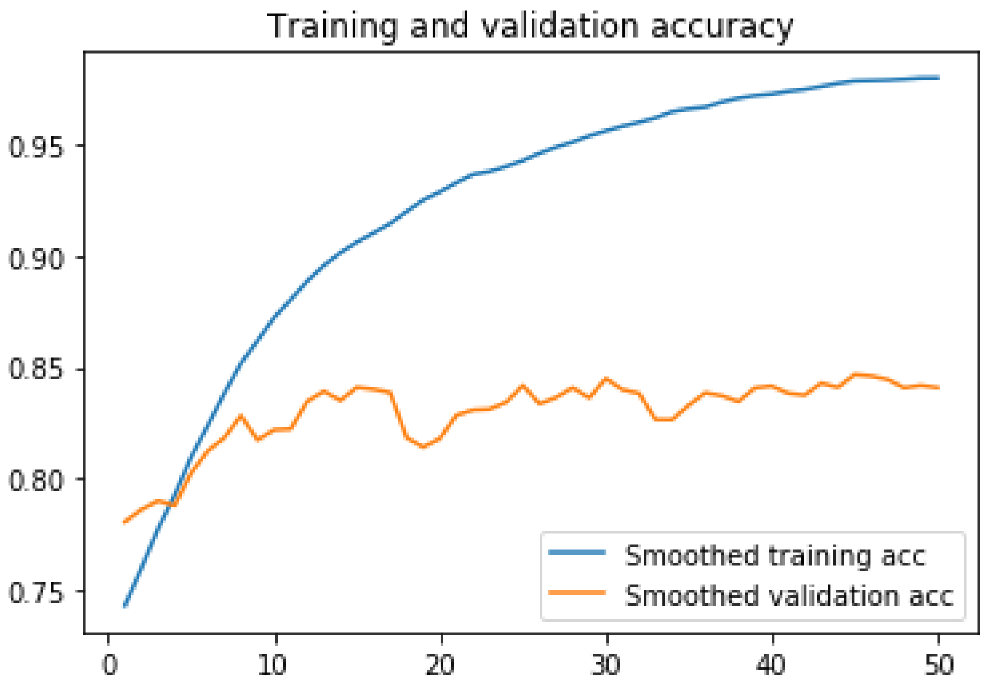
[21]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



###Smoothed plots

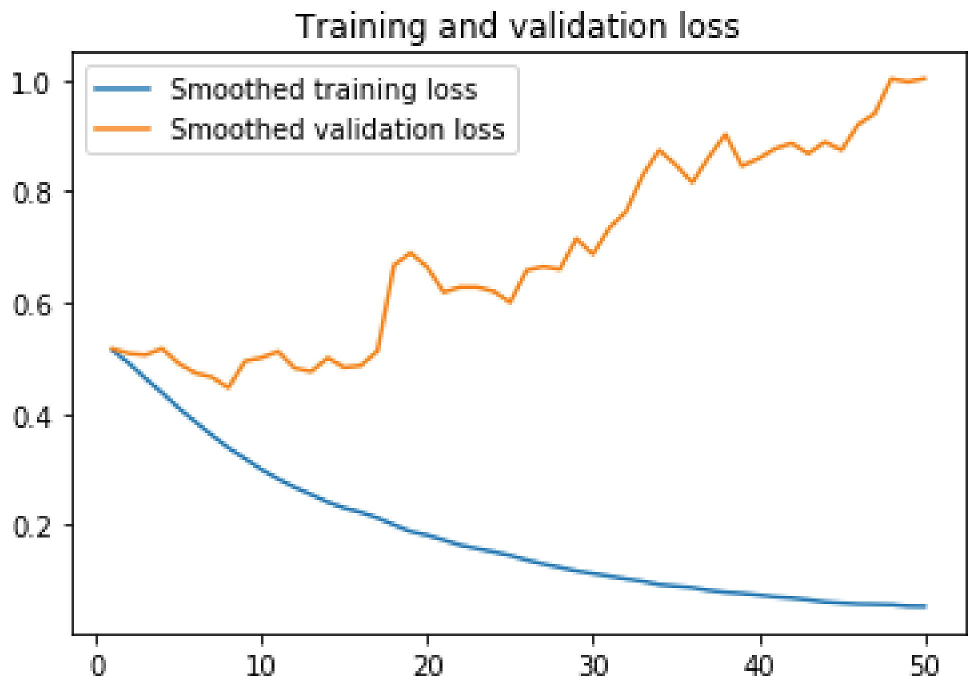
[22]:
```python
import matplotlib.pyplot as plt
import numpy as np

def smooth_curve(points, factor=0.8):
  smoothed_points = []
  for point in points:
    if smoothed_points:
      previous = smoothed_points[-1]
      smoothed_points.append(previous * factor + point * (1 - factor))
    else:
      smoothed_points.append(point)
  return smoothed_points

acc = history.history['acc']
val_acc = history.history['val_acc']
epoch = range(1, len(acc) + 1)

plt.plot(epoch, smooth_curve(acc), 'tab:blue', label='Smoothed training acc')
plt.plot(epoch, smooth_curve(val_acc), 'tab:orange', label='Smoothed validation␣
 ↪acc')
plt.title('Training and validation accuracy')
plt.legend(loc='lower right')
plt.figure()
```

[22]: `<Figure size 432x288 with 0 Axes>`



```
<Figure size 432x288 with 0 Axes>
```

[23]:
```
loss = history.history['loss']
val_loss = history.history['val_loss']
epoch = range(1, len(loss) + 1)

plt.plot(epoch, smooth_curve(loss), 'tab:blue', label='Smoothed training loss')
plt.plot(epoch, smooth_curve(val_loss), 'tab:orange', label='Smoothed␣
 ↪validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```


Training and validation loss
Smoothed training loss
Smoothed validation loss
1.0
0.8
0.6
0.4
0.2
0
10
20
30
40
50

cars_act_CNN_final

October 18, 2019

1 Visualizing activation for cars CNN model

[100]:
```
from google.colab import drive, files
drive.mount('/content/gdrive')
gdrive_path = 'gdrive/My\ Drive/Colab\ Notebooks/'
!rsync -ah --progress\
    {gdrive_path}cars_dl/*.zip\
    {gdrive_path}model_weights.h5\
    {gdrive_path}model_structure.json\
    '/content'
!unzip -qo '*.zip'
!rm *.zip
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call
drive.mount("/content/gdrive", force_remount=True).
sending incremental file list
cars-20191002T050118Z-001.zip
        490.68M 100%  142.05MB/s    0:00:03 (xfr#1, to-chk=2/3)
```

[0]:
```
from keras.models import model_from_json
from pathlib import Path
from keras.preprocessing import image
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
import os
```

##Load the json file that contains the model's structure

[0]:
```
f = Path("model_structure.json")
model_structure = f.read_text()
```

1.1 Recreate the Keras model object from the json data

[0]:
```
model = model_from_json(model_structure)
```

1.2 Re-load the model's trained weights

[0]:
```
model.load_weights("model_weights.h5")
```

1.3 Summary of model

[105]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 126, 126, 32)      896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 63, 63, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 61, 61, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 30, 30, 32)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 30, 30, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 32)        9248
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
dropout_3 (Dropout)          (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 12, 12, 32)        9248
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 6, 6, 32)          0
_________________________________________________________________
dropout_4 (Dropout)          (None, 6, 6, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               147584
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 258
=================================================================
Total params: 176,482
Trainable params: 176,482
```

```
Non-trainable params: 0
_________________________________________________________________
```

1.4 Preprocessing a single image

[106]:
```python
from keras.preprocessing.image import ImageDataGenerator
import os

test_data_dir = 'cars/test_set/'
testCarImages = os.listdir('cars/test_set/car/')
img_path = 'cars/test_set/car/' + testCarImages[60]

test_datagen = ImageDataGenerator(rescale = 1./255)
test_set = test_datagen.flow_from_directory(test_data_dir,
                                            target_size = (128, 128),
                                            batch_size = 32,
                                            class_mode = 'categorical',
                                            shuffle=True,
                                            seed=42)
```

```
Found 260 images belonging to 2 classes.
```

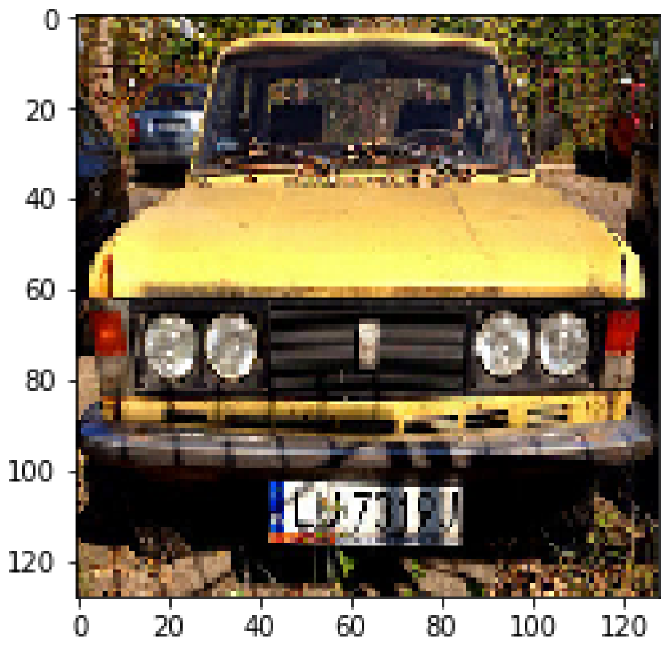
[107]:
```python
# We preprocess the image into a 4D tensor
from keras.preprocessing import image
import numpy as np
img = image.load_img(img_path, target_size=(128, 128))
img_tensor = image.img_to_array(img)
img_tensor = np.expand_dims(img_tensor, axis=0)
# Remember that the model was trained on inputs
# that were preprocessed in the following way:
img_tensor /= 255.
# Its shape is (1, 128, 128, 3)
print(img_tensor.shape)
```

```
(1, 128, 128, 3)
```

##Displaying the test image

[108]:
```python
plt.imshow(img_tensor[0])
```

[108]:
```
<matplotlib.image.AxesImage at 0x7eff8de020b8>
```



##Instantiating a model from an input tensor and a list of otput tensors

[0]:
```
from keras import models
# Extracts the outputs of the top 12 layers:
layer_outputs = [layer.output for layer in model.layers[:2]]
# Creates a model that will return these outputs, given the model input:
activation_model = models.Model(inputs=model.input, outputs=layer_outputs)
```

##Prediction

[110]:
```
testCarImages = os.listdir('cars/test_set/car/')
testFilename = 'cars/test_set/car/' + testCarImages[60]

test_image = image.load_img(testFilename, target_size = (128, 128))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis=0)
result = model.predict(test_image)
test_set.class_indices
if result[0][0] < 0.3:
  prediction = '[0.1]'
elif result[0][0] > 0.7:
  prediction = '[1.0]'
else:
  prediction = 'Unsure'


dimage = Image.open(testFilename)
dimage
```

[110]:

[111]:
```
print("Prediction: " + str(result[0][0]))
```

```
Prediction: 1.0
```

[0]:
```
# This will return a list of 5 Numpy arrays: one array per layer activation
activations = activation_model.predict(img_tensor)
```

First entry in the outputs: the output of the first layer of the original model

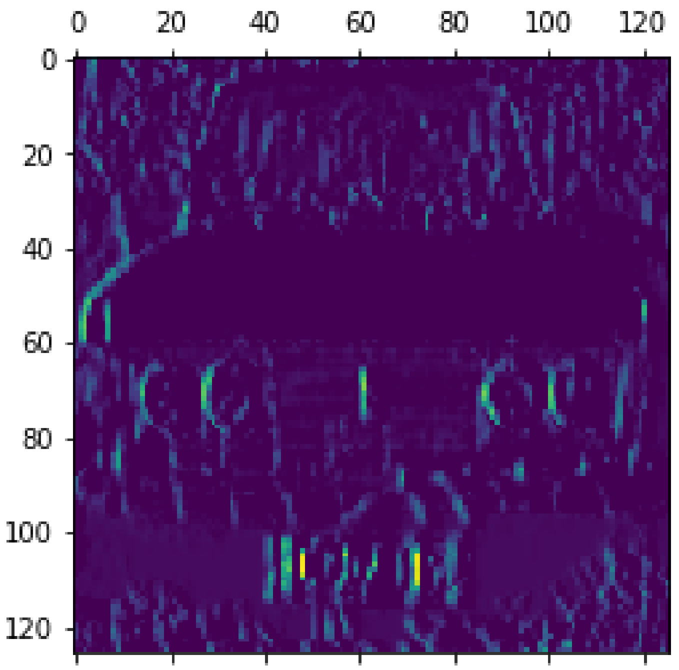
[113]:
```
first_layer_activation = activations[0]
print(first_layer_activation.shape)
```

```
(1, 126, 126, 32)
```

##Plotting the 5th channel of the activation of the first layer of the original model

[114]:
```
import matplotlib.pyplot as plt
plt.matshow(first_layer_activation[0, :, :, 5], cmap='viridis')
```

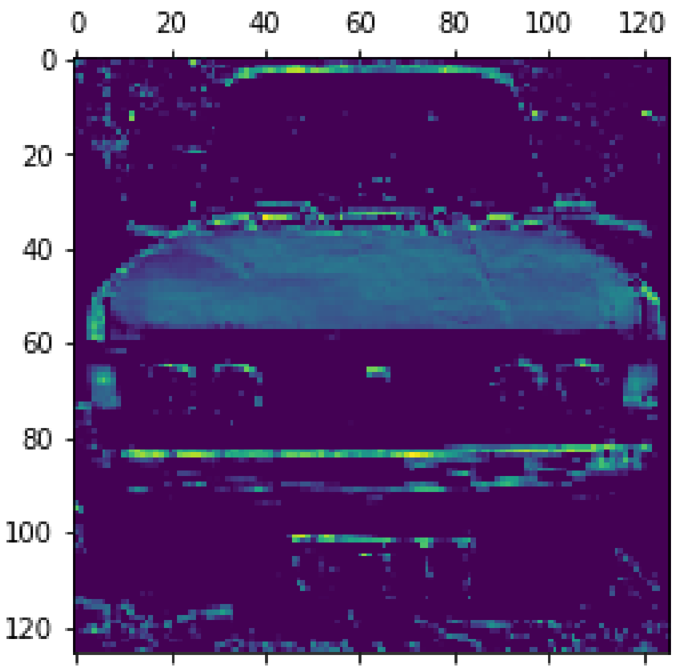
[114]:
```
<matplotlib.image.AxesImage at 0x7eff8e0c8358>
```



1.5 Plot the 20th channel of the activation of the first layer of the model

[129]:
```
plt.matshow(first_layer_activation[0, :, :, 20], cmap='viridis')
```

[129]:
```
<matplotlib.image.AxesImage at 0x7eff8cfd54e0>
```



1.6 Visualizing every channel in every intermediate activation

[117]:
```
print(layer_names)
```

```
['conv2d_1', 'max_pooling2d_1', 'dropout_1', 'conv2d_2', 'max_pooling2d_2',
'dropout_2', 'conv2d_3', 'max_pooling2d_3']
```

1.6.1 Instantiating a model from an input tensor and a list of output tensors

[0]:
```
layer_outputs = [layer.output for layer in model.layers[:12]]
# Extracts the outputs of the top 12 layers
activation_model = models.Model(inputs=model.input, outputs=layer_outputs) #␣
↪Creates a model that will return these outputs, given the model input
```

##Running the model in predict mode

[0]:
```
activations = activation_model.predict(img_tensor)
# Returns a list of five Numpy arrays: one array per layer activation
```

##Activation of first layer

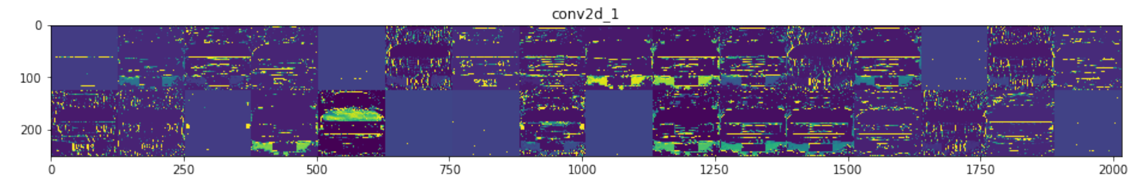
[120]:
```
first_layer_activation = activations[0]
print(first_layer_activation.shape)
```

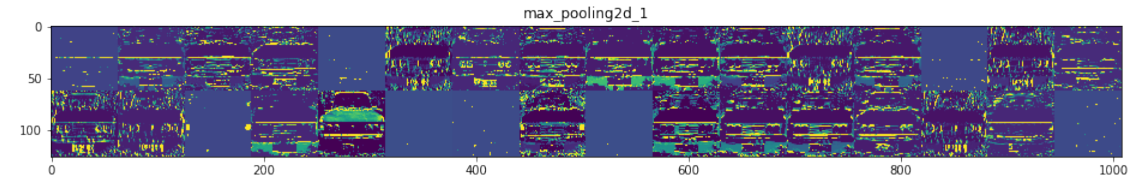
```
(1, 126, 126, 32)
```

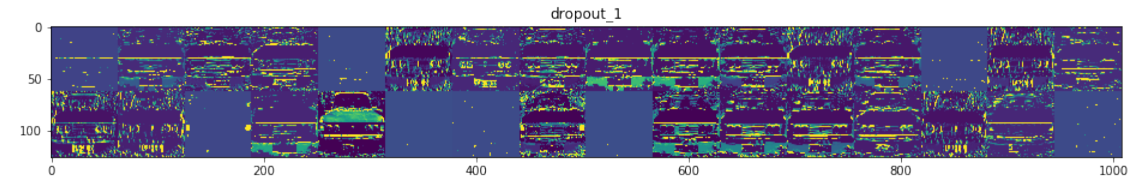
##Visualise every layer

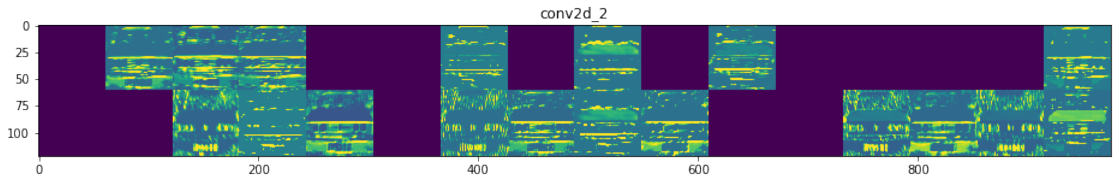
[122]:
```
layer_names = []
for layer in model.layers[:12]:
    layer_names.append(layer.name)                                  # Names of␣
↪the layers, so you can have them as part of your plot

images_per_row = 16

for layer_name, layer_activation in zip(layer_names, activations):  # Displays␣
↪the feature maps
    n_features = layer_activation.shape[-1]                         # Number of␣
↪features in the feature map
    size = layer_activation.shape[1]                                # The␣
↪feature map has shape (1, size, size, n_features).
    n_cols = n_features // images_per_row                           # Tiles the␣
↪activation channels in this matrix
    display_grid = np.zeros((size * n_cols, images_per_row * size))
    for col in range(n_cols):                                       # Tiles␣
↪each filter into a big horizontal grid
        for row in range(images_per_row):
            channel_image = layer_activation[0,
                                             :, :,
                                             col * images_per_row + row]
            channel_image -= channel_image.mean()                  #␣
↪Post-processes the feature to make it visually palatable
            channel_image /= channel_image.std()
            channel_image *= 64
            channel_image += 128
            channel_image = np.clip(channel_image, 0, 255).astype('uint8')
            display_grid[col * size : (col + 1) * size,            # Displays␣
↪the grid
                         row * size : (row + 1) * size] = channel_image
    scale = 1. / size
    plt.figure(figsize=(scale * display_grid.shape[1],
                        scale * display_grid.shape[0]))
    plt.title(layer_name)
    plt.grid(False)
    plt.imshow(display_grid, aspect='auto', cmap='viridis')
```

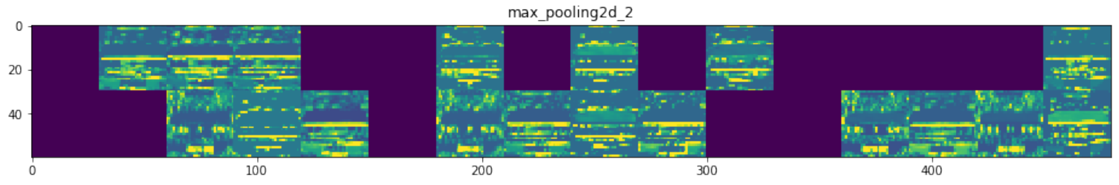
```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:18: RuntimeWarning:
invalid value encountered in true_divide
```

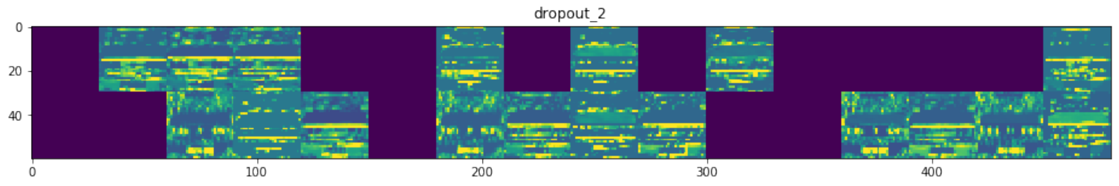


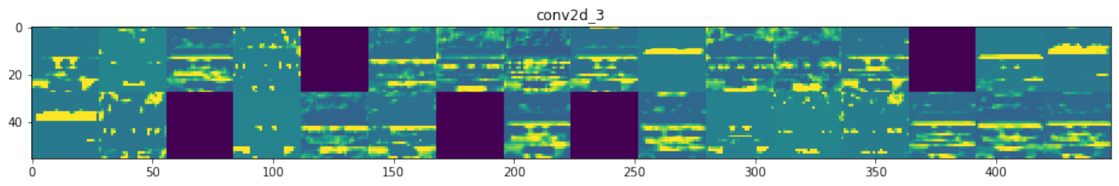


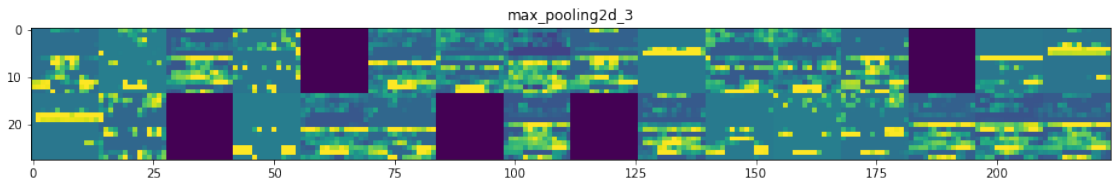


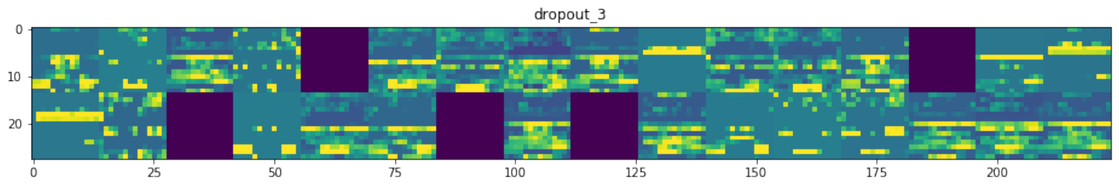

conv2d_2
0
25
50
75
100
0
200
400
600
800

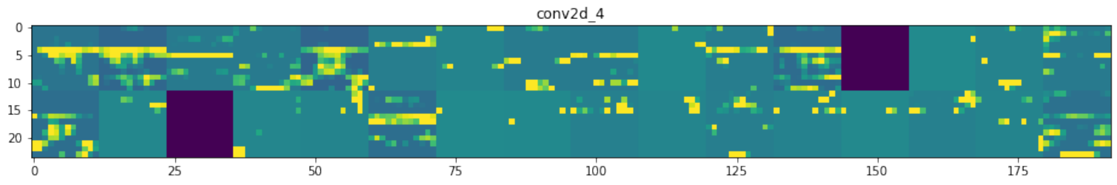

max_pooling2d_2
0
20
40
0
100
200
300
400

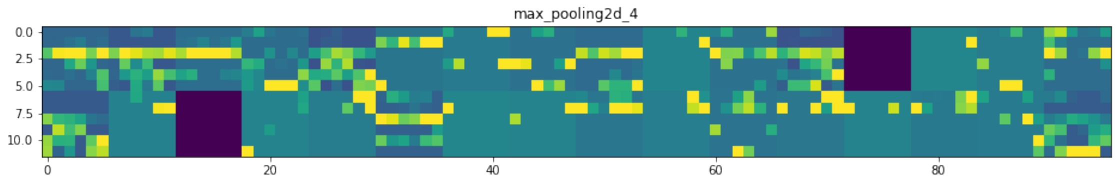

dropout_2
0
20
40
0
100
200
300
400

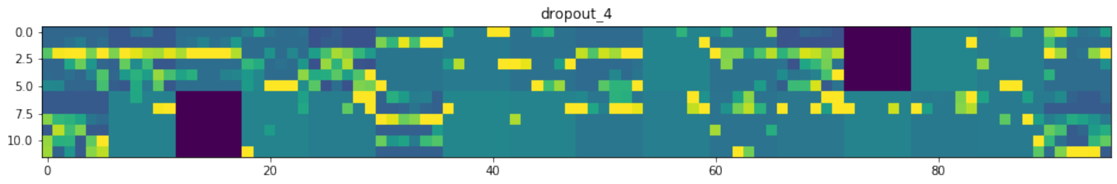

conv2d_3
0
20
40
0
50
100
150
200
250
300
350
400


max_pooling2d_3
0
10
20
0
25
50
75
100
125
150
175
200


dropout_3
0
10
20
0
25
50
75
100
125
150
175
200







[123]:
```
#Method to print several images in a single row
def plots(img, figsize=(12,6), rows = 1, titles = 1):

  if type(img[0]) is np.ndarray:
    img = np.array(img).astype(np.float_)
  if (img.shape[-1] != 3):
    img = img.transpose((0, 2, 3, 1))
  f =plt.figure(figsize = figsize)
  cols = 7//rows if (len(img) % 2 == 0) else len(img)//rows + 1

  for i in range(cols):
    sp = f.add_subplot(rows, cols, i+1)
    sp.axis('On')

    result = model.predict(img)
    print(result[i][0])
    test_set.class_indices
    if result[i][0] < 0.3:
      prediction = '[0.1]'
    elif result[i][0] > 0.7:
      prediction = '[1.0]'
    else:
      prediction = 'Unsure'

    if titles is not None:
      sp.set_title(titles[i], fontsize = 10, pad = 10)
      sp.set_xlabel('Prediction: ' + prediction)
    plt.imshow(img[i], interpolation = None if np.interp else 'none')


imgs, labels = next(test_set)
plots(imgs, titles = labels)
```

```
0.99815184
0.93519294
0.06446395
0.19790655
0.10372881
0.05639749
0.16553427

/usr/local/lib/python3.6/dist-packages/matplotlib/text.py:1191: FutureWarning:
elementwise comparison failed; returning scalar instead, but in the future will
perform elementwise comparison
  if s != self._text:
```