Deep Learning Project Report

Group Members:

Name	Student No.	Section
Kunal Naresh Kumar Thapar	1107686	COMP-5413-WA
Rahul Niraj Singh	1099198	COMP-5413-WA

Task 1:
Object-centric Image recognition task

Dataset:

CIFAR10, Caltech101, Caltech256

Deep Learning Project Report

CIFAR-10

Introduction:

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Dataset Link: https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz

Model:

ResNet50 model with weights pre-trained on

ImageNet. ResNet50 Architecture:

ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks. This model was the winner of ImageNet challenge in 2015. The fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+layers successfully. Prior to ResNet training very deep neural networks was difficult due to the problem of vanishing gradients.

We have used autoencoder for feature extraction and data augmentation for training efficiently

Source Code:

```
# -*- coding: utf-8 -*-
"""Cifar10Resnet50.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1qL0CMLSIEhYwlfglFqzIA0RyqKBlN8AQ
"""

# Commented out IPython magic to ensure Python compatibility.
# %tensorflow_version 1.x
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input
import tensorflow.keras as keras
```

Deep Learning Project Report

```
from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras import optimizers
import tensorflow as tf
from keras.utils import np_utils
from keras.models import load_model
from keras.datasets import cifar10
from keras.preprocessing import image
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import cv2



"""**Importing Libraries**"""

resnetModel = ResNet50(weights='imagenet', include_top=False, input_shape=(200, 200, 3))

"""**Importing resnet 50 model**"""

(trainInput , trainLabel) , (testInput, testLabel) = cifar10.load_data()

trainInput = trainInput / 255.0
testInput = testInput / 255.0

"""**Getting cifar 10 data**"""

trainLabel = np_utils.to_categorical(trainLabel, 10)
testLabel = np_utils.to_categorical(testLabel, 10)

"""**converting label to one hot encoding**"""

model = models.Sequential()

#encoder
model.add(layers.Conv2D(16, (3, 3), activation='relu', padding='same'))
model.add(layers.MaxPooling2D((2, 2), padding='same') )
# model.add(layers.Conv2D(8, (3, 3), activation='relu', padding='same') )
# model.add(layers.MaxPooling2D((2, 2), padding='same') )
# model.add(layers.Conv2D(8, (3, 3), activation='relu', padding='same') )
# model.add(layers.MaxPooling2D((2, 2), padding='same') )

#decoder
# model.add(layers.Conv2D(8, (3, 3), activation='relu', padding='same') )
# model.add(layers.UpSampling2D((2, 2)) )
# model.add (layers.Conv2D(8, (3, 3), activation='relu', padding='same') )
# model.add(layers.UpSampling2D((2, 2)) )
model.add(layers.Conv2D(16, (3, 3), activation='relu', padding='same'))
model.add(layers.UpSampling2D((2, 2)) )
```

Deep Learning Project Report

```
model.add(layers.Conv2D(1, (3, 3), activation='sigmoid', padding='same') )

model.compile(optimizer='adadelta', loss='binary_crossentropy')

"""**Feature extraction model**"""

model.fit(trainInput, trainInput,
          epochs=50,
          batch_size=128,
          shuffle=True,
          validation_data=(testInput, testInput))

"""**Training autoencoder**"""

model = models.Sequential()

#upsampling to increase image....
model.add(layers.UpSampling2D((2,2)))
model.add(layers.UpSampling2D((2,2)))
model.add(layers.UpSampling2D((2,2)))

model.add(resnetModel)

model.add(layers.Flatten())
model.add(layers.BatchNormalization())
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.BatchNormalization())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.BatchNormalization())
model.add(layers.Dense(10, activation='softmax'))

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

"""**Preparing model with the help of resnet 50 pretrained model with imagenet**"""

from keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(horizontal_flip = True)

"""**data augmentation**"""

import time
start = time.time()
history = model.fit_generator(datagen.flow(trainInput, trainLabel,batch_size = 20), epochs=5,
validation_data=(testInput, testLabel), steps_per_epoch = len(trainInput) // 20)
end = time.time()
```

Deep Learning Project Report

```
print("Total Training time: ",(end-start))

"""**Training time**"""

import keras
from matplotlib import 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', 'val'], loc='upper left')
plt.show()
plt.savefig("performance_cifar10.png")

"""**Analyzing performance**

**training model**
"""

score = model.evaluate(testInput, testLabel)
print('The model achieved a accuracy of %.2f%%.' % (score[1]*100))

"""**evaluating model performance**"""
```

Output:

Run 1:

```
Epoch 1/5
2499/2500 [============================>.] - ETA: 0s - loss: 0.2511 - acc:
0.9000Epoch 1/5
10000/2500
[==============================================================================
==========================================] - 43s 4ms/sample - loss:
0.1177 - acc: 0.9623
2500/2500 [==============================] - 868s 347ms/step - loss: 0.2511
- acc: 0.9000 - val_loss: 0.1088 - val_acc: 0.9623
Epoch 2/5
2499/2500 [============================>.] - ETA: 0s - loss: 0.1572 - acc:
0.9383Epoch 1/5
10000/2500
[==============================================================================
==========================================] - 42s 4ms/sample - loss:
0.0669 - acc: 0.9776
2500/2500 [==============================] - 849s 340ms/step - loss: 0.1573
- acc: 0.9383 - val_loss: 0.0688 - val_acc: 0.9776
Epoch 3/5
```

Deep Learning Project Report

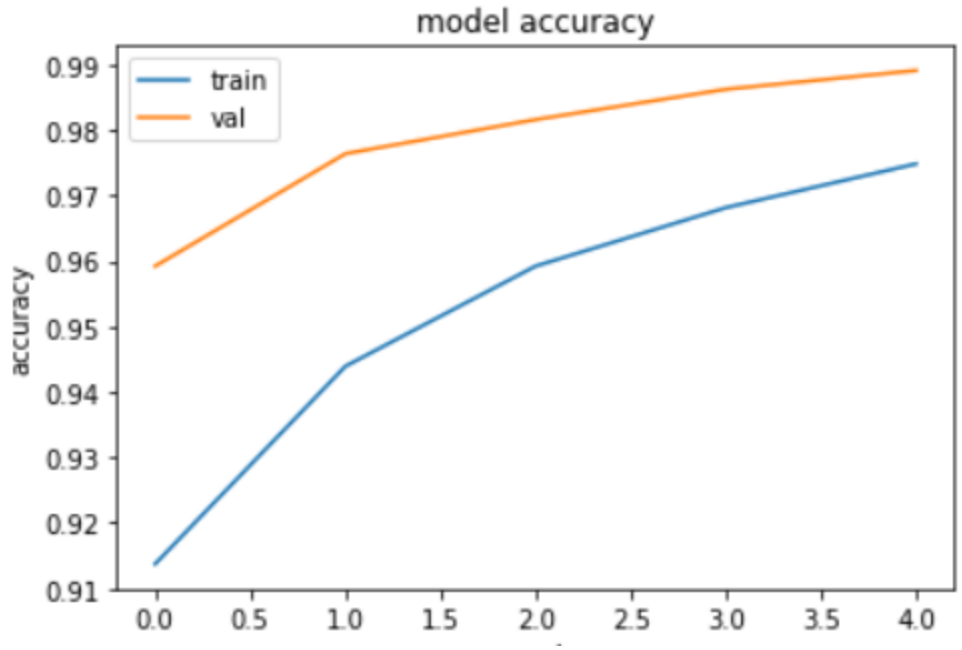
```
2499/2500 [============================>.] - ETA: 0s - loss: 0.1187 - acc:
0.9577Epoch 1/5
10000/2500
[==============================================================================
==========================================] - 42s 4ms/sample - loss:
0.0457 - acc: 0.9839
2500/2500 [==============================] - 851s 340ms/step - loss: 0.1187
- acc: 0.9577 - val_loss: 0.0518 - val_acc: 0.9839
Epoch 4/5
2499/2500 [============================>.] - ETA: 0s - loss: 0.0960 - acc:
0.9677Epoch 1/5
10000/2500
[==============================================================================
==========================================] - 42s 4ms/sample - loss:
0.0502 - acc: 0.9863
2500/2500 [==============================] - 856s 342ms/step - loss: 0.0960
- acc: 0.9677 - val_loss: 0.0433 - val_acc: 0.9863
Epoch 5/5
2499/2500 [============================>.] - ETA: 0s - loss: 0.0791 - acc:
0.9748Epoch 1/5
10000/2500
[==============================================================================
==========================================] - 42s 4ms/sample - loss:
0.0293 - acc: 0.9896
2500/2500 [==============================] - 857s 343ms/step - loss: 0.0791
- acc: 0.9748 - val_loss: 0.0335 - val_acc: 0.9896
```

Output Run 1:

```
[12] score = model.evaluate(testInput, testLabel)
     print('The model achieved a accuracy of %.2f%%.' % (score[1]*100))

     10000/10000 [==============================] - 41s 4ms/sample - loss: 0.0335 - acc: 0.9896
     The model achieved a accuracy of 98.96%.
```

Deep Learning Project Report



Run 2 :

```
 Epoch 1/5
2499/2500 [============================>.] - ETA: 0s - loss: 0.2459 -
acc: 0.9021Epoch 1/5
10000/2500
[==============================================================================
==============================================] - 59s 6ms/sample -
loss: 0.1103 - acc: 0.9592
2500/2500 [==============================] - 1198s 479ms/step - loss:
0.2459 - acc: 0.9021 - val_loss: 0.1154 - val_acc: 0.9592
Epoch 2/5
2499/2500 [============================>.] - ETA: 0s - loss: 0.1576 -
acc: 0.9390Epoch 1/5
10000/2500
[==============================================================================
==============================================] - 59s 6ms/sample -
loss: 0.0736 - acc: 0.9764
2500/2500 [==============================] - 1185s 474ms/step - loss:
0.1576 - acc: 0.9390 - val_loss: 0.0726 - val_acc: 0.9764
Epoch 3/5
2499/2500 [============================>.] - ETA: 0s - loss: 0.1207 -
acc: 0.9564Epoch 1/5
10000/2500
[==============================================================================
==============================================] - 58s 6ms/sample -
loss: 0.0568 - acc: 0.9816
2500/2500 [==============================] - 1188s 475ms/step - loss:
0.1207 - acc: 0.9564 - val_loss: 0.0565 - val_acc: 0.9816
Epoch 4/5
2499/2500 [============================>.] - ETA: 0s - loss: 0.0982 -
acc: 0.9666Epoch 1/5
10000/2500
[==============================================================================
```

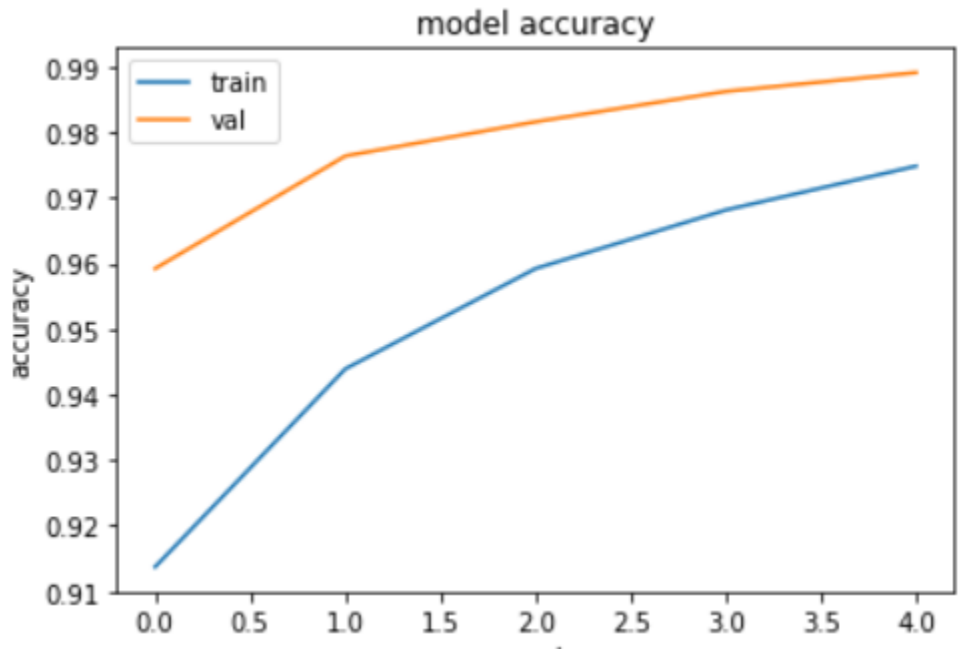
Deep Learning Project Report

```
==============================================] - 58s 6ms/sample -
loss: 0.0408 - acc: 0.9862
2500/2500 [==============================] - 1184s 473ms/step - loss:
0.0982 - acc: 0.9666 - val_loss: 0.0431 - val_acc: 0.9862
Epoch 5/5
2499/2500 [============================>.] - ETA: 0s - loss: 0.0817 -
acc: 0.9740Epoch 1/5
10000/2500
[==============================================================================
==============================================] - 58s 6ms/sample -
loss: 0.0351 - acc: 0.9891
2500/2500 [==============================] - 1183s 473ms/step - loss:
0.0817 - acc: 0.9740 - val_loss: 0.0339 - val_acc: 0.9891
```

Output Run 2:

```
score = model.evaluate(testInput, testLabel)
print('The model achieved a accuracy of %.2f%%.' % (score[1]*100))
```

```
10000/10000 [==============================] - 57s 6ms/sample - loss: 0.0339 - acc: 0.9891
The model achieved a accuracy of 98.91%.
```



Run 3:

```
Epoch 1/5
2499/2500 [============================>.] - ETA: 0s - loss: 0.2476 - acc:
0.9003Epoch 1/5
10000/2500
[==============================================================================
==============================================] - 44s 4ms/sample - loss:
0.1085 - acc: 0.9613
```

Deep Learning Project Report

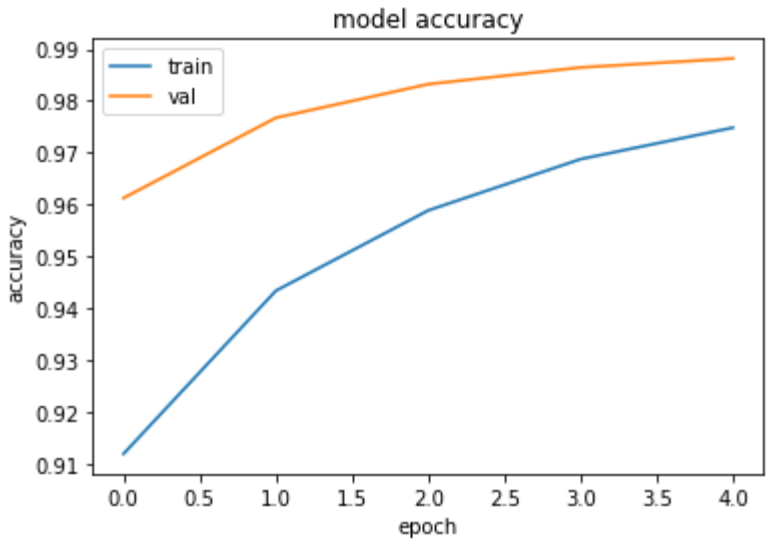
```
2500/2500 [==============================] - 875s 350ms/step - loss: 0.2476
- acc: 0.9003 - val_loss: 0.1085 - val_acc: 0.9613
Epoch 2/5
2499/2500 [==============================>.] - ETA: 0s - loss: 0.1573 - acc:
0.9381Epoch 1/5
10000/2500
[==============================================================================
==============================================] - 42s 4ms/sample - loss:
0.0655 - acc: 0.9767
2500/2500 [==============================] - 855s 342ms/step - loss: 0.1573
- acc: 0.9381 - val_loss: 0.0701 - val_acc: 0.9767
Epoch 3/5
2499/2500 [==============================>.] - ETA: 0s - loss: 0.1200 - acc:
0.9562Epoch 1/5
10000/2500
[==============================================================================
==============================================] - 43s 4ms/sample - loss:
0.0467 - acc: 0.9832
2500/2500 [==============================] - 858s 343ms/step - loss: 0.1200
- acc: 0.9562 - val_loss: 0.0515 - val_acc: 0.9832
Epoch 4/5
2499/2500 [==============================>.] - ETA: 0s - loss: 0.0967 - acc:
0.9670Epoch 1/5
10000/2500
[==============================================================================
==============================================] - 42s 4ms/sample - loss:
0.0356 - acc: 0.9865
2500/2500 [==============================] - 857s 343ms/step - loss: 0.0967
- acc: 0.9670 - val_loss: 0.0418 - val_acc: 0.9865
Epoch 5/5
2499/2500 [==============================>.] - ETA: 0s - loss: 0.0805 - acc:
0.9742Epoch 1/5
10000/2500
[==============================================================================
==============================================] - 42s 4ms/sample - loss:
0.0368 - acc: 0.9882
2500/2500 [==============================] - 857s 343ms/step - loss: 0.0804
- acc: 0.9742 - val_loss: 0.0357 - val_acc: 0.9882
```

Output Run 3:

```
[12] score = model.evaluate(testInput, testLabel)
     print('The model achieved a accuracy of %.2f%%.' % (score[1]*100))

     10000/10000 [==============================] - 41s 4ms/sample - loss: 0.0357 - acc: 0.9882
     The model achieved a accuracy of 98.82%.
```

Deep Learning Project Report




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

Result:

	First Run	Second Run	Third Run	Average
Accuracy	98.96%	98.91%	98.82%	98.89%
Training Time	4281.315982s	5939.951s	4300.386s	4840.939s

Deep Learning Project Report

CALTECH-101

Introduction:

Pictures of objects belonging to 101 categories. About 40 to 800 images per category. Most categories have about 50 images. Collected in September 2003 by Fei-Fei Li, Marco Andreetto, and Marc 'Aurelio Ranzato. The size of each image is roughly 300 x 200 pixels.

Dataset Link: http://www.vision.caltech.edu/Image_Datasets/Caltech101/#Download

Model:

ResNet50 model with weights pre-trained on

ImageNet. ResNet50 Architecture:

ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks. This model was the winner of ImageNet challenge in 2015. The fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+layers successfully. Prior to ResNet training very deep neural networks was difficult due to the problem of vanishing gradients.

We have used autoencoder for feature extraction and data augmentation for training efficiently

Source Code:

```
# -*- coding: utf-8 -*-
"""Caltech101Resnet50.ipynb

Automatically generated by Colaboratory.

Original file is located at
https://colab.research.google.com/drive/1bKYwFCIY9ehr0uBSlqnMHNx3MBSutlkl

**Downloading dataset**
"""

!wget http://www.vision.caltech.edu/Image_Datasets/Caltech101/101_ObjectCategories.tar.gz

"""**Unzipping dataset file**"""

!tar -xf 101_ObjectCategories.tar.gz

!rm -rf '101_ObjectCategories/BACKGROUND_Google'

# Commented out IPython magic to ensure Python compatibility.
# %tensorflow_version 1.x
from keras.preprocessing.image import ImageDataGenerator

"""**In each folder, split 30 images to training set and remaining to test set**"""
```

Deep Learning Project Report

```
import os
import math

os.mkdir("caltech_test") # stores test data

for cat in os.listdir("101_ObjectCategories/"):
# moves x portion of images per category into test images
os.mkdir("caltech_test/"+cat) # new category folder
imgs = os.listdir("101_ObjectCategories/"+cat) # all image filenames
test_imgs = imgs[30:len(imgs)]
for t_img in test_imgs: # move test portion
os.rename("101_ObjectCategories/"+cat+"/"+t_img, "caltech_test/"+cat+"/"+t_img)

def fixed_generator(generator):
for batch in generator:
yield (batch, batch)

"""**Preparing data for feature extraction**"""

# import cv2
# training_images = []
# validation_images = []
# for root, dirs, files in os.walk('101_ObjectCategories/'):
#     for file in files:
#         # print(file)
#         # print(root + '/' + file)
#         image = cv2.imread(root + '/' + file)
#         dim = (256, 256)
#         # resize image
#         image = cv2.resize(image, dim, interpolation = cv2.INTER_AREA)
#         training_images.append(image)


# for root, dirs, files in os.walk('caltech_test/'):
#     for file in files:
#         image = cv2.imread(root + '/' + file)
#         dim = (256, 256)
#         # resize image
#         image = cv2.resize(image, dim, interpolation = cv2.INTER_AREA)
#         validation_images.append(image)




from keras.applications.resnet50 import preprocess_input

train_gen_autoencoder = ImageDataGenerator(rescale=1./255) #rotation_range = 30, zoom_range =
0.20,
# fill_mode = "nearest", shear_range = 0.20, horizontal_flip = True,
# width_shift_range = 0.1, height_shift_range = 0.1)
```

```
train_flow = train_gen_autoencoder.flow_from_directory("101_ObjectCategories/",
target_size=(256, 256), batch_size=64,class_mode=None)
valid_flow = train_gen_autoencoder.flow_from_directory("caltech_test/", target_size=(256, 256),
batch_size=64,class_mode=None)

"""**Importing necessary library**"""

from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input
import tensorflow.keras as keras
from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras import optimizers
import tensorflow as tf
from keras.utils import np_utils
from keras.models import load_model
from keras.datasets import cifar10
from keras.preprocessing import image
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import cv2

"""**Autoencoder model**"""

model = models.Sequential()

#encoder
# model.add(layers.Conv2D(256, (3, 3), activation='relu', padding='same'))
# model.add(layers.MaxPooling2D((2, 2), padding='same') )
# model.add(layers.Conv2D(128, (3, 3), activation='relu', padding='same'))
# model.add(layers.MaxPooling2D((2, 2), padding='same') )
# model.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same'))
# model.add(layers.MaxPooling2D((2, 2), padding='same') )
# model.add(layers.Conv2D(32, (3, 3), activation='relu', padding='same'))
# model.add(layers.MaxPooling2D((2, 2), padding='same') )
model.add(layers.Conv2D(16, (3, 3), activation='relu', padding='same'))
model.add(layers.MaxPooling2D((2, 2), padding='same') )
# model.add(layers.Conv2D(8, (3, 3), activation='relu', padding='same') )
# model.add(layers.MaxPooling2D((2, 2), padding='same') )
# model.add(layers.Conv2D(8, (3, 3), activation='relu', padding='same') )
# model.add(layers.MaxPooling2D((2, 2), padding='same') )

#decoder
# model.add(layers.Conv2D(8, (3, 3), activation='relu', padding='same') )
# model.add(layers.UpSampling2D((2, 2)) )
# model.add (layers.Conv2D(8, (3, 3), activation='relu', padding='same') )
# model.add(layers.UpSampling2D((2, 2)) )
model.add(layers.Conv2D(16, (3, 3), activation='relu', padding='same'))
model.add(layers.UpSampling2D((2, 2)) )
# model.add(layers.Conv2D(32, (3, 3), activation='relu', padding='same'))
```

```
# model.add(layers.UpSampling2D((2, 2)) )
# model.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same'))
# model.add(layers.UpSampling2D((2, 2)) )
# model.add(layers.Conv2D(128, (3, 3), activation='relu', padding='same'))
# model.add(layers.UpSampling2D((2, 2)) )
# model.add(layers.Conv2D(256, (3, 3), activation='relu', padding='same'))
# model.add(layers.UpSampling2D((2, 2)) )

model.add(layers.Conv2D(1, (3, 3), activation='sigmoid', padding='same') )

model.compile(optimizer='adadelta', loss='binary_crossentropy')

"""**Training autoencoder**"""

# model.fit(training_images, training_images,
#               epochs=5,
#               batch_size=128,
#               shuffle=True,
#               validation_data=(validation_images, validation_images))
model.fit_generator(fixed_generator(train_flow), epochs=5,steps_per_epoch=3030//128,
validation_steps=5647//128,validation_data = fixed_generator(valid_flow) )

"""**data augmentation and genearting data for trainig with keras flow from directory**"""

from keras.applications.resnet50 import preprocess_input

train_gen = ImageDataGenerator(validation_split=0.2, preprocessing_function=preprocess_input)
#rotation_range = 30, zoom_range = 0.20,
# fill_mode = "nearest", shear_range = 0.20, horizontal_flip = True,
# width_shift_range = 0.1, height_shift_range = 0.1)
train_flow = train_gen.flow_from_directory("101_ObjectCategories/", target_size=(256, 256),
batch_size=32, subset="training")
valid_flow = train_gen.flow_from_directory("101_ObjectCategories/", target_size=(256, 256),
batch_size=32, subset="validation")

test_gen = ImageDataGenerator(preprocessing_function=preprocess_input)
test_flow = test_gen.flow_from_directory("caltech_test", target_size=(256, 256), batch_size=32)

"""**Preparing model with the help of resnet 50 pretrained model on imagenet**"""

from keras.applications.resnet50 import ResNet50
from keras.layers import GlobalAveragePooling2D, BatchNormalization, Dropout, Dense
from keras.models import Model,Sequential


res = ResNet50(weights='imagenet', include_top=False, input_shape=(256, 256, 3))
# model = Sequential()

# for layer in res.layers[:-1]:
#     model.add(layer)
```

Deep Learning Project Report

```
for layer in res.layers: # freezing the layers
layer.trainable = False

# model.add(GlobalAveragePooling2D())
# model.add(BatchNormalization())
# model.add(Dropout(0.5))
# model.add(Dense(512, activation='relu'))
# model.add(BatchNormalization())
# model.add(Dropout(0.5))
# model.add(Dense(101, activation='softmax'))


x = res.output # get the output from the loaded model
x = GlobalAveragePooling2D()(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
x = Dense(512, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
x = Dense(101, activation='softmax')(x)

model = Model(res.input, x)

model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['accuracy']) # compile the model - we're training using the Adam Optimizer and Categorical Cross Entropy as the loss function

model.summary()

"""**Training model**"""

import time
start = time.time()
history = model.fit_generator(train_flow, epochs=15, validation_data=valid_flow)
end = time.time()

print("Total Training time: ",(end-start))

import keras
from matplotlib import 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', 'val'], loc='upper left')
plt.show()
plt.savefig("performance_caltech101.png")
```

Deep Learning Project Report

"""**Evaluating model**"""

score = model.evaluate(test_flow)

print('The model achieved a accuracy of %.2f%%.' % (score[1]*100))

output:

Run 1:

```
 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 1/15
76/76 [==============================] - 12s 159ms/step - loss: 3.1010 -
acc: 0.3407 - val_loss: 0.6620 - val_acc: 0.8267
Epoch 2/15
76/76 [==============================] - 8s 107ms/step - loss: 0.8624 - acc:
0.7845 - val_loss: 0.4403 - val_acc: 0.8713
Epoch 3/15
76/76 [==============================] - 8s 107ms/step - loss: 0.5074 - acc:
0.8651 - val_loss: 0.4158 - val_acc: 0.8746
Epoch 4/15
76/76 [==============================] - 8s 108ms/step - loss: 0.3425 - acc:
0.9127 - val_loss: 0.3976 - val_acc: 0.8878
Epoch 5/15
76/76 [==============================] - 8s 108ms/step - loss: 0.2693 - acc:
0.9242 - val_loss: 0.3584 - val_acc: 0.8894
Epoch 6/15
76/76 [==============================] - 8s 108ms/step - loss: 0.2094 - acc:
0.9437 - val_loss: 0.3615 - val_acc: 0.8911
Epoch 7/15
76/76 [==============================] - 8s 108ms/step - loss: 0.1489 - acc:
0.9668 - val_loss: 0.3582 - val_acc: 0.8927
Epoch 8/15
76/76 [==============================] - 8s 107ms/step - loss: 0.1376 - acc:
0.9652 - val_loss: 0.3632 - val_acc: 0.8927
Epoch 9/15
76/76 [==============================] - 8s 108ms/step - loss: 0.1036 - acc:
0.9770 - val_loss: 0.3434 - val_acc: 0.8977
Epoch 10/15
76/76 [==============================] - 8s 107ms/step - loss: 0.0938 - acc:
0.9762 - val_loss: 0.3315 - val_acc: 0.9026
Epoch 11/15
76/76 [==============================] - 8s 107ms/step - loss: 0.0962 - acc:
0.9811 - val_loss: 0.3366 - val_acc: 0.9092
Epoch 12/15
76/76 [==============================] - 8s 108ms/step - loss: 0.0778 - acc:
0.9801 - val_loss: 0.3430 - val_acc: 0.9010
Epoch 13/15
```

Deep Learning Project Report

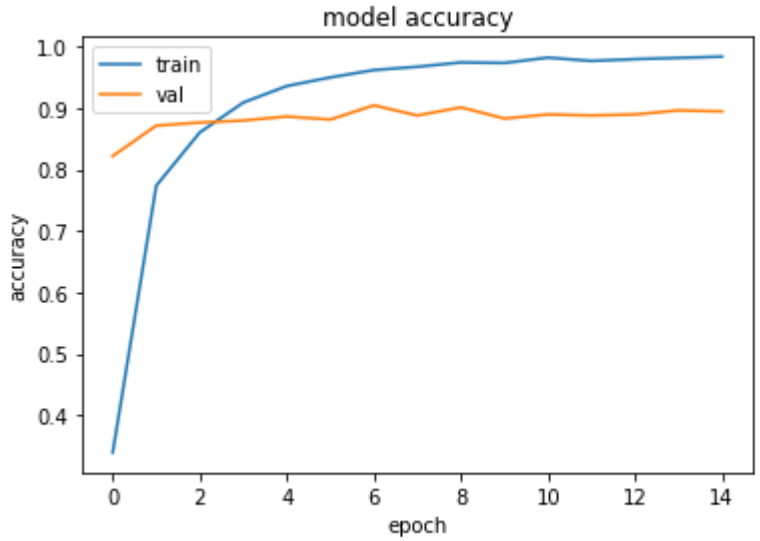
```
76/76 [==============================] - 8s 108ms/step - loss: 0.0769 - acc:
0.9836 - val_loss: 0.3714 - val_acc: 0.8861
Epoch 14/15
76/76 [==============================] - 8s 108ms/step - loss: 0.0757 - acc:
0.9823 - val_loss: 0.3711 - val_acc: 0.9010
Epoch 15/15
76/76 [==============================] - 8s 107ms/step - loss: 0.0683 - acc:
0.9805 - val_loss: 0.3610 - val_acc: 0.8894
```

Output Run 1:

```
[17] score = model.evaluate(test_flow)

     print('The model achieved a accuracy of %.2f%%.' % (score[1]*100))
```

```
177/177 [==============================] - 17s 95ms/step
The model achieved a accuracy of 92.47%.
```




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

Run 2 :

```
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 1/15
76/76 [==============================] - 11s 149ms/step - loss: 3.0807 -
acc: 0.3444 - val_loss: 0.6780 - val_acc: 0.8234
Epoch 2/15
```

Deep Learning Project Report

```
76/76 [==============================] - 8s 106ms/step - loss: 0.8566 - acc:
0.7810 - val_loss: 0.4743 - val_acc: 0.8729
Epoch 3/15
76/76 [==============================] - 8s 105ms/step - loss: 0.5005 - acc:
0.8710 - val_loss: 0.4038 - val_acc: 0.8795
Epoch 4/15
76/76 [==============================] - 8s 105ms/step - loss: 0.3545 - acc:
0.9012 - val_loss: 0.3935 - val_acc: 0.8861
Epoch 5/15
76/76 [==============================] - 8s 106ms/step - loss: 0.2621 - acc:
0.9341 - val_loss: 0.3602 - val_acc: 0.8878
Epoch 6/15
76/76 [==============================] - 8s 106ms/step - loss: 0.2095 - acc:
0.9457 - val_loss: 0.3433 - val_acc: 0.8993
Epoch 7/15
76/76 [==============================] - 8s 106ms/step - loss: 0.1599 - acc:
0.9616 - val_loss: 0.3771 - val_acc: 0.8878
Epoch 8/15
76/76 [==============================] - 8s 106ms/step - loss: 0.1345 - acc:
0.9711 - val_loss: 0.3534 - val_acc: 0.8960
Epoch 9/15
76/76 [==============================] - 8s 106ms/step - loss: 0.1226 - acc:
0.9711 - val_loss: 0.3484 - val_acc: 0.8993
Epoch 10/15
76/76 [==============================] - 8s 106ms/step - loss: 0.1075 - acc:
0.9712 - val_loss: 0.3434 - val_acc: 0.9010
Epoch 11/15
76/76 [==============================] - 8s 106ms/step - loss: 0.0905 - acc:
0.9772 - val_loss: 0.3654 - val_acc: 0.8977
Epoch 12/15
76/76 [==============================] - 8s 106ms/step - loss: 0.0806 - acc:
0.9794 - val_loss: 0.3575 - val_acc: 0.8911
Epoch 13/15
76/76 [==============================] - 8s 106ms/step - loss: 0.0689 - acc:
0.9819 - val_loss: 0.3581 - val_acc: 0.8927
Epoch 14/15
76/76 [==============================] - 8s 106ms/step - loss: 0.0733 - acc:
0.9803 - val_loss: 0.3685 - val_acc: 0.9043
Epoch 15/15
76/76 [==============================] - 8s 106ms/step - loss: 0.0779 - acc:
0.9823 - val_loss: 0.3867 - val_acc: 0.8944
```

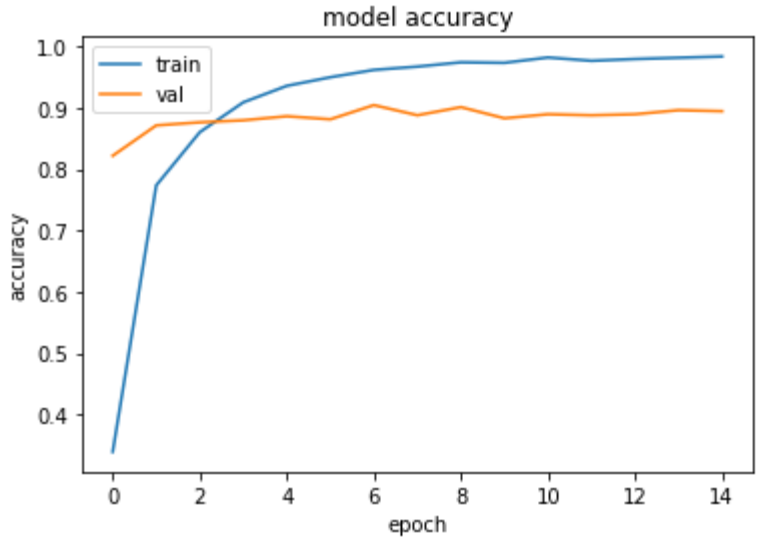
Output Run 2:

```
[17] score = model.evaluate(test_flow)

     print('The model achieved a accuracy of %.2f%%.' % (score[1]*100))

     177/177 [==============================] - 16s 89ms/step
     The model achieved a accuracy of 92.58%.
```

Deep Learning Project Report



<Figure size 432x288 with 0 Axes>

Run 3:

Epoch 1/15
76/76 [==============================] - 33s 435ms/step - loss: 3.0280 - acc: 0.3603 - val_loss: 0.7808 - val_acc: 0.7921
Epoch 2/15
76/76 [==============================] - 28s 366ms/step - loss: 0.8796 - acc: 0.7725 - val_loss: 0.5601 - val_acc: 0.8399
Epoch 3/15
76/76 [==============================] - 28s 367ms/step - loss: 0.5054 - acc: 0.8668 - val_loss: 0.4835 - val_acc: 0.8647
Epoch 4/15
76/76 [==============================] - 28s 369ms/step - loss: 0.3419 - acc: 0.9137 - val_loss: 0.4622 - val_acc: 0.8729
Epoch 5/15
76/76 [==============================] - 28s 367ms/step - loss: 0.2517 - acc: 0.9349 - val_loss: 0.4145 - val_acc: 0.8795
Epoch 6/15
76/76 [==============================] - 28s 370ms/step - loss: 0.1961 - acc: 0.9513 - val_loss: 0.4261 - val_acc: 0.8812
Epoch 7/15
76/76 [==============================] - 28s 367ms/step - loss: 0.1692 - acc: 0.9601 - val_loss: 0.3908 - val_acc: 0.8746
Epoch 8/15
76/76 [==============================] - 28s 368ms/step - loss: 0.1341 - acc: 0.9696 - val_loss: 0.3893 - val_acc: 0.8795
Epoch 9/15
76/76 [==============================] - 28s 370ms/step - loss: 0.1090 - acc: 0.9756 - val_loss: 0.3875 - val_acc: 0.8894
Epoch 10/15
76/76 [==============================] - 28s 367ms/step - loss: 0.0988 - acc: 0.9790 - val_loss: 0.3903 - val_acc: 0.8927
Epoch 11/15

Deep Learning Project Report

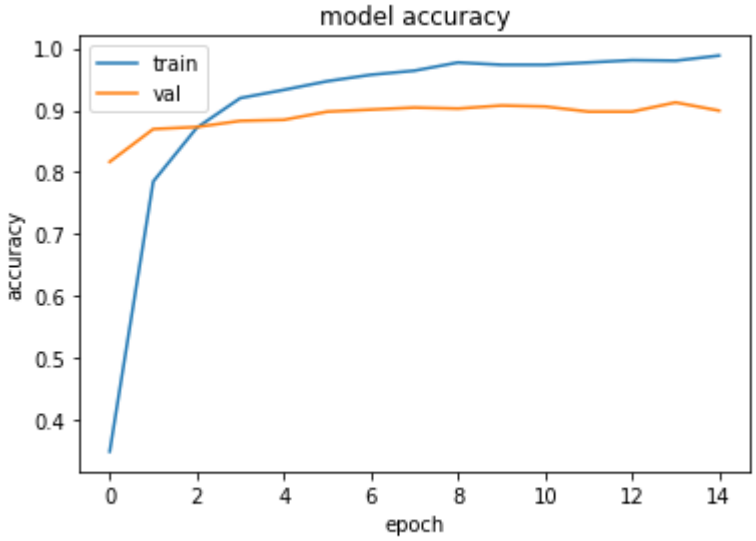
76/76 [==============================] - 28s 367ms/step - loss: 0.0917 - acc: 0.9774 - val_loss: 0.3721 - val_acc: 0.8944
Epoch 12/15
76/76 [==============================] - 28s 373ms/step - loss: 0.0749 - acc: 0.9851 - val_loss: 0.3600 - val_acc: 0.9010
Epoch 13/15
76/76 [==============================] - 28s 371ms/step - loss: 0.0726 - acc: 0.9840 - val_loss: 0.3747 - val_acc: 0.9026
Epoch 14/15
76/76 [==============================] - 28s 368ms/step - loss: 0.0605 - acc: 0.9868 - val_loss: 0.3547 - val_acc: 0.9125
Epoch 15/15
76/76 [==============================] - 28s 367ms/step - loss: 0.0658 - acc: 0.9799 - val_loss: 0.3487 - val_acc: 0.9059
<keras.callbacks.History at 0x7fa33447ac88>

Output Run 3:

```
[ ]  result = model.evaluate(test_flow)

     print('The model achieved a loss of %.2f and accuracy of %.2f%%.' % (result[0], result[1]*100))

     177/177 [==============================] - 56s 316ms/step
     The model achieved a loss of 0.23 and accuracy of 93.38%.
```




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

Result:

	First Run	Second Run	Third Run	Average
Accuracy	92.83%	92.58%	93.38%	**92.93%**
Training Time	**126.615982s**	**124.316127s**	**128.22113s**	**126.38s**

Deep Learning Project Report

CALTECH-256

Introduction:

Caltech-256 dataset have 257 categories with 30608 total pictures.

Dataset Link:
http://www.vision.caltech.edu/Image_Datasets/Caltech256/256_ObjectCategories.tar

Model:

ResNet50 model with weights pre-trained on

ImageNet. ResNet50 Architecture:

ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks. This model was the winner of ImageNet challenge in 2015. The fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+layers successfully. Prior to ResNet training very deep neural networks was difficult due to the problem of vanishing gradients.

We have used autoencoder for feature extraction and data augmentation for training efficiently

Source Code:

```
# -*- coding: utf-8 -*-

"""Caltech256Resnet50.ipynb


Automatically generated by Colaboratory.


Original file is located at

    https://colab.research.google.com/drive/1dFZvmt_YYG5LAM95nTLo77fmsuTz847Z


**Downloading dataset**

"""


!wget http://www.vision.caltech.edu/Image_Datasets/Caltech256/256_ObjectCategories.tar
```

Deep Learning Project Report

```
"""**Unzipping dataset file**"""

!tar -xf 256_ObjectCategories.tar

!rm -rf '256_ObjectCategories/056.dog/greg' # remove random image folder
!rm -rf '256_ObjectCategories/198.spider/RENAME2' #remove random files

# Commented out IPython magic to ensure Python compatibility.
# %tensorflow_version 1.x
from keras.preprocessing.image import ImageDataGenerator

"""**In each folder, split 30 images to training set and remaining to test set**"""

import os
import math

os.mkdir("caltech_test") # stores test data

for cat in os.listdir("256_ObjectCategories/"):
  # moves x portion of images per category into test images
  os.mkdir("caltech_test/"+cat) # new category folder
  imgs = os.listdir("256_ObjectCategories/"+cat) # all image filenames
  # split = math.floor(len(imgs)*TEST_SPLIT)
  test_imgs = imgs[30:len(imgs)]
  for t_img in test_imgs: # move test portion
    os.rename("256_ObjectCategories/"+cat+"/"+t_img, "caltech_test/"+cat+"/"+t_img)

def fixed_generator(generator):
    for batch in generator:
        yield (batch, batch)
```

Deep Learning Project Report

"""**Preparing data for feature extraction**"""

```
# import cv2
from keras.applications.resnet50 import preprocess_input

train_gen_autoencoder = ImageDataGenerator(rescale=1./255) #rotation_range = 30,
zoom_range = 0.20,
# fill_mode = "nearest", shear_range = 0.20, horizontal_flip = True,
# width_shift_range = 0.1, height_shift_range = 0.1)
train_flow = train_gen_autoencoder.flow_from_directory("256_ObjectCategories/",
target_size=(256, 256), batch_size=64,class_mode=None)
valid_flow = train_gen_autoencoder.flow_from_directory("caltech_test/", target_size=(256,
256), batch_size=64,class_mode=None)
```

"""**Importing necessary library**"""

```
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input
import tensorflow.keras as keras
from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras import optimizers
import tensorflow as tf
from keras.utils import np_utils
from keras.models import load_model
from keras.datasets import cifar10
from keras.preprocessing import image
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import cv2
```

Deep Learning Project Report

"""**Autoencoder model**"""

```
model = models.Sequential()

#encoder
# model.add(layers.Conv2D(256, (3, 3), activation='relu', padding='same'))
# model.add(layers.MaxPooling2D((2, 2), padding='same') )
# model.add(layers.Conv2D(128, (3, 3), activation='relu', padding='same'))
# model.add(layers.MaxPooling2D((2, 2), padding='same') )
# model.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same'))
# model.add(layers.MaxPooling2D((2, 2), padding='same') )
# model.add(layers.Conv2D(32, (3, 3), activation='relu', padding='same'))
# model.add(layers.MaxPooling2D((2, 2), padding='same') )
model.add(layers.Conv2D(16, (3, 3), activation='relu', padding='same'))
model.add(layers.MaxPooling2D((2, 2), padding='same') )
# model.add(layers.Conv2D(8, (3, 3), activation='relu', padding='same') )
# model.add(layers.MaxPooling2D((2, 2), padding='same') )
# model.add(layers.Conv2D(8, (3, 3), activation='relu', padding='same') )
# model.add(layers.MaxPooling2D((2, 2), padding='same') )

#decoder
# model.add(layers.Conv2D(8, (3, 3), activation='relu', padding='same') )
# model.add(layers.UpSampling2D((2, 2)) )
# model.add (layers.Conv2D(8, (3, 3), activation='relu', padding='same') )
# model.add(layers.UpSampling2D((2, 2)) )
model.add(layers.Conv2D(16, (3, 3), activation='relu', padding='same'))
model.add(layers.UpSampling2D((2, 2)) )
# model.add(layers.Conv2D(32, (3, 3), activation='relu', padding='same'))
# model.add(layers.UpSampling2D((2, 2)) )
```

Deep Learning Project Report

```
# model.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same'))
# model.add(layers.UpSampling2D((2, 2)) )
# model.add(layers.Conv2D(128, (3, 3), activation='relu', padding='same'))
# model.add(layers.UpSampling2D((2, 2)) )
# model.add(layers.Conv2D(256, (3, 3), activation='relu', padding='same'))
# model.add(layers.UpSampling2D((2, 2)) )

model.add(layers.Conv2D(1, (3, 3), activation='sigmoid', padding='same') )

model.compile(optimizer='adadelta', loss='binary_crossentropy')

"""**Training autoencoder**"""

# model.fit(training_images, training_images,
#           epochs=5,
#           batch_size=128,
#           shuffle=True,
#           validation_data=(validation_images, validation_images))
model.fit_generator(fixed_generator(train_flow), epochs=5,steps_per_epoch=7710//128,
validation_steps=22897//128,validation_data = fixed_generator(valid_flow) )

"""**data augmentation and generating data for training with keras flow from directory**"""

from keras.applications.resnet50 import preprocess_input # make sure to match original model's
preprocessing function

# train_gen = ImageDataGenerator(validation_split=0.2,
preprocessing_function=preprocess_input,rotation_range = 30, zoom_range = 0.20,
# fill_mode = "nearest", shear_range = 0.20, horizontal_flip = True,
# width_shift_range = 0.1, height_shift_range = 0.1)
```

Deep Learning Project Report

```
# train_gen = ImageDataGenerator(validation_split=0.2,
preprocessing_function=preprocess_input,rotation_range = 30, zoom_range = 0.20,

# fill_mode = "nearest", shear_range = 0.20, horizontal_flip = True,

# width_shift_range = 0.1, height_shift_range = 0.1)


train_gen = ImageDataGenerator(validation_split=0.2,
preprocessing_function=preprocess_input)

train_flow = train_gen.flow_from_directory("256_ObjectCategories/", target_size=(256, 256),
batch_size=128, subset="training")

valid_flow = train_gen.flow_from_directory("256_ObjectCategories/", target_size=(256, 256),
batch_size=128, subset="validation")


test_gen = ImageDataGenerator(preprocessing_function=preprocess_input)

test_flow = test_gen.flow_from_directory("caltech_test", target_size=(256, 256), batch_size=32)


"""**Preparing model with the help of resnet 50 pretrained model on imagenet**"""


from keras.applications.resnet50 import ResNet50

from keras.layers import GlobalAveragePooling2D, BatchNormalization, Dropout, Dense

from keras.models import Model



res = ResNet50(weights='imagenet', include_top=False, input_shape=(256, 256, 3)) # load resnet
model, with pretrained imagenet weights.


for layer in res.layers: # because our finetuning dataset is similar to the imagenet dataset, we can
freeze the convolutional layers

  layer.trainable = False


x = res.output # get the output from the loaded model

x = GlobalAveragePooling2D()(x)

x = BatchNormalization()(x)
```

Deep Learning Project Report

```
x = Dropout(0.5)(x)

x = Dense(512, activation='relu')(x)

x = BatchNormalization()(x)

x = Dropout(0.5)(x)

x = Dense(257, activation='softmax')(x)


model = Model(res.input, x) # create the model, setting input/output


model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['accuracy']) # compile the model - we're training using the Adam Optimizer and Categorical Cross Entropy as the loss function




model.summary() # prints the structure of our model


"""**Training model**"""


import time

start = time.time()

history = model.fit_generator(train_flow, epochs=15, validation_data=valid_flow)

end = time.time()


print("Total Training time: ",(end-start))


import keras

from matplotlib import pyplot as plt


plt.plot(history.history['acc'])

plt.plot(history.history['val_acc'])

plt.title('model accuracy')

plt.ylabel('accuracy')
```

Deep Learning Project Report

```
plt.xlabel('epoch')

plt.legend(['train', 'val'], loc='upper left')

plt.show()

plt.savefig("performance_caltech256.png")


"""**Evaluating model**"""


score = model.evaluate(test_flow)


print('The model achieved an accuracy of %.2f%%.' % (score[1]*100))
```

Output:

Run 1:

```
Epoch 1/15
49/49 [==============================] - 33s 664ms/step - loss: 4.8029 - acc:
0.1321 - val_loss: 2.1711 - val_acc: 0.5564
Epoch 2/15
49/49 [==============================] - 22s 446ms/step - loss: 2.1793 - acc:
0.5216 - val_loss: 1.3736 - val_acc: 0.6933
Epoch 3/15
49/49 [==============================] - 23s 478ms/step - loss: 1.4201 - acc:
0.6795 - val_loss: 1.1510 - val_acc: 0.7237
Epoch 4/15
49/49 [==============================] - 23s 475ms/step - loss: 1.0704 - acc:
0.7459 - val_loss: 1.0570 - val_acc: 0.7380
Epoch 5/15
49/49 [==============================] - 23s 478ms/step - loss: 0.8219 - acc:
0.8034 - val_loss: 1.0033 - val_acc: 0.7399
Epoch 6/15
49/49 [==============================] - 23s 469ms/step - loss: 0.6794 - acc:
0.8370 - val_loss: 0.9827 - val_acc: 0.7419
Epoch 7/15
49/49 [==============================] - 24s 491ms/step - loss: 0.5510 - acc:
0.8684 - val_loss: 0.9595 - val_acc: 0.7536
Epoch 8/15
49/49 [==============================] - 24s 496ms/step - loss: 0.4795 - acc:
0.8800 - val_loss: 0.9382 - val_acc: 0.7685
Epoch 9/15
49/49 [==============================] - 23s 478ms/step - loss: 0.4095 - acc:
0.9034 - val_loss: 0.9291 - val_acc: 0.7607
Epoch 10/15
49/49 [==============================] - 24s 481ms/step - loss: 0.3536 - acc:
0.9100 - val_loss: 0.9200 - val_acc: 0.7646
Epoch 11/15
49/49 [==============================] - 23s 471ms/step - loss: 0.3109 - acc:
0.9234 - val_loss: 0.9149 - val_acc: 0.7659
Epoch 12/15
49/49 [==============================] - 24s 483ms/step - loss: 0.2711 - acc:
0.9340 - val_loss: 0.9078 - val_acc: 0.7678
Epoch 13/15
```

Deep Learning Project Report

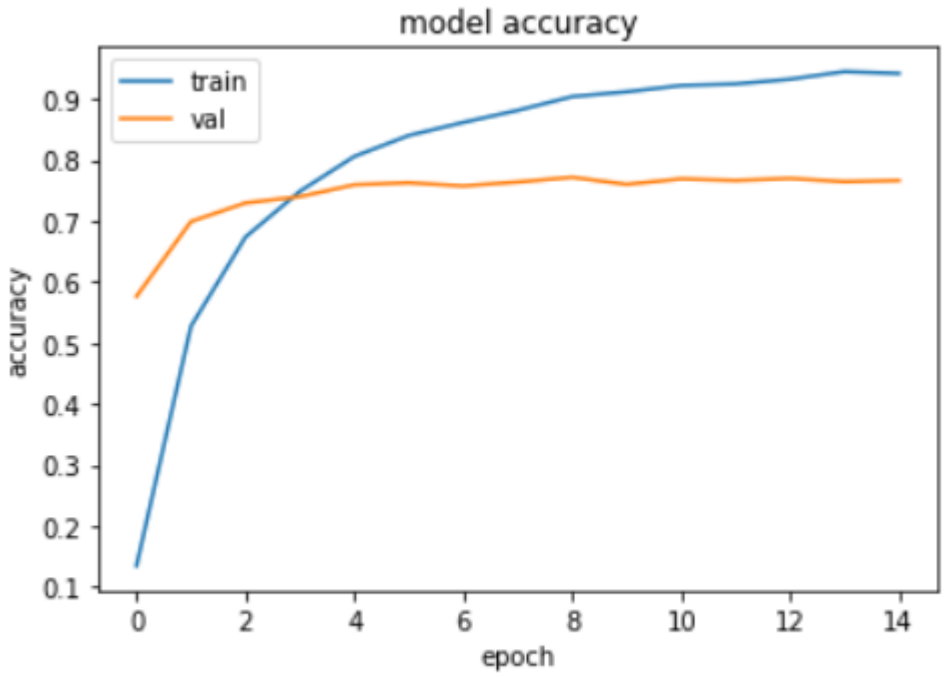
```
49/49 [==============================] - 23s 472ms/step - loss: 0.2399 - acc:
0.9419 - val_loss: 0.8962 - val_acc: 0.7711
Epoch 14/15
49/49 [==============================] - 23s 479ms/step - loss: 0.2254 - acc:
0.9461 - val_loss: 0.8996 - val_acc: 0.7782
Epoch 15/15
49/49 [==============================] - 24s 486ms/step - loss: 0.2073 - acc:
0.9482 - val_loss: 0.9112 - val_acc: 0.7717
<keras.callbacks.History at 0x7fe0d79be3c8>
```

Output Run 1:

```
[ ]  # from PIL import ImageFile
     # ImageFile.LOAD_TRUNCATED_IMAGES = True
     result = model.evaluate(test_flow)

     print('The model achieved a loss of %.2f and accuracy of %.2f%%.' % (result[0], result[1]*100))

     179/179 [==============================] - 82s 456ms/step
     The model achieved a loss of 0.95 and accuracy of 77.42%.
```




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

Run 2:

```
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 1/15
```

Deep Learning Project Report

```
49/49 [==============================] - 32s 659ms/step - loss: 4.7579 -
acc: 0.1427 - val_loss: 2.1185 - val_acc: 0.5746
Epoch 2/15
49/49 [==============================] - 25s 520ms/step - loss: 2.1927 -
acc: 0.5211 - val_loss: 1.4061 - val_acc: 0.6958
Epoch 3/15
49/49 [==============================] - 26s 537ms/step - loss: 1.4252 -
acc: 0.6723 - val_loss: 1.1863 - val_acc: 0.7231
Epoch 4/15
49/49 [==============================] - 26s 532ms/step - loss: 1.0958 -
acc: 0.7408 - val_loss: 1.0920 - val_acc: 0.7367
Epoch 5/15
49/49 [==============================] - 26s 530ms/step - loss: 0.8521 -
acc: 0.7890 - val_loss: 1.0238 - val_acc: 0.7549
Epoch 6/15
49/49 [==============================] - 27s 541ms/step - loss: 0.6766 -
acc: 0.8405 - val_loss: 0.9999 - val_acc: 0.7555
Epoch 7/15
49/49 [==============================] - 27s 547ms/step - loss: 0.5656 -
acc: 0.8609 - val_loss: 0.9858 - val_acc: 0.7542
Epoch 8/15
49/49 [==============================] - 26s 535ms/step - loss: 0.4836 -
acc: 0.8780 - val_loss: 0.9763 - val_acc: 0.7549
Epoch 9/15
49/49 [==============================] - 26s 533ms/step - loss: 0.4200 -
acc: 0.8925 - val_loss: 0.9623 - val_acc: 0.7691
Epoch 10/15
49/49 [==============================] - 26s 536ms/step - loss: 0.3745 -
acc: 0.9083 - val_loss: 0.9549 - val_acc: 0.7646
Epoch 11/15
49/49 [==============================] - 26s 537ms/step - loss: 0.3391 -
acc: 0.9135 - val_loss: 0.9430 - val_acc: 0.7594
Epoch 12/15
49/49 [==============================] - 27s 550ms/step - loss: 0.2848 -
acc: 0.9347 - val_loss: 0.9224 - val_acc: 0.7691
Epoch 13/15
49/49 [==============================] - 26s 533ms/step - loss: 0.2545 -
acc: 0.9392 - val_loss: 0.9389 - val_acc: 0.7717
Epoch 14/15
49/49 [==============================] - 26s 536ms/step - loss: 0.2363 -
acc: 0.9394 - val_loss: 0.9525 - val_acc: 0.7626
Epoch 15/15
49/49 [==============================] - 26s 530ms/step - loss: 0.2005 -
acc: 0.9515 - val_loss: 0.9445 - val_acc: 0.7665
```

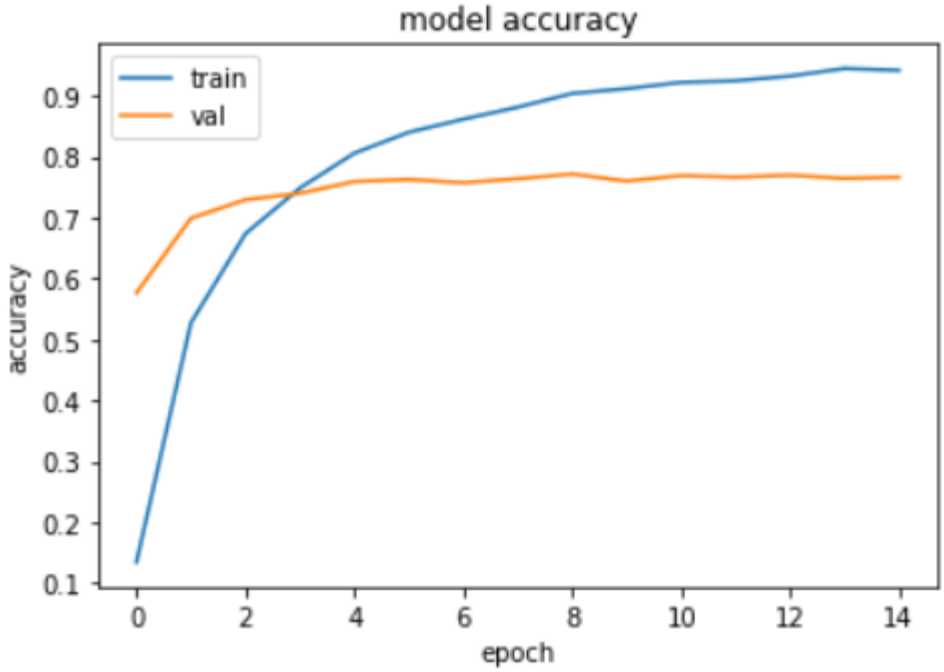
Output Run 2:

```
[17] score = model.evaluate(test_flow)

     print('The model achieved a accuracy of %.2f%%.' % (score[1]*100))
```

```
716/716 [==============================] - 96s 135ms/step
The model achieved a accuracy of 76.76%.
```

Deep Learning Project Report



<Figure size 432x288 with 0 Axes>

Run 3:

```
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 1/15
49/49 [==============================] - 32s 647ms/step - loss: 4.7531 -
acc: 0.1358 - val_loss: 2.1474 - val_acc: 0.5765
Epoch 2/15
49/49 [==============================] - 25s 515ms/step - loss: 2.1497 -
acc: 0.5276 - val_loss: 1.3908 - val_acc: 0.6991
Epoch 3/15
49/49 [==============================] - 27s 547ms/step - loss: 1.4133 -
acc: 0.6722 - val_loss: 1.1608 - val_acc: 0.7296
Epoch 4/15
49/49 [==============================] - 28s 562ms/step - loss: 1.0526 -
acc: 0.7487 - val_loss: 1.0584 - val_acc: 0.7399
Epoch 5/15
49/49 [==============================] - 27s 542ms/step - loss: 0.8251 -
acc: 0.8064 - val_loss: 1.0048 - val_acc: 0.7594
Epoch 6/15
49/49 [==============================] - 27s 547ms/step - loss: 0.6790 -
acc: 0.8395 - val_loss: 0.9772 - val_acc: 0.7626
Epoch 7/15
49/49 [==============================] - 27s 554ms/step - loss: 0.5780 -
acc: 0.8592 - val_loss: 0.9566 - val_acc: 0.7575
Epoch 8/15
```

Deep Learning Project Report

```
49/49 [==============================] - 27s 547ms/step - loss: 0.4851 -
acc: 0.8793 - val_loss: 0.9495 - val_acc: 0.7639
Epoch 9/15
49/49 [==============================] - 27s 542ms/step - loss: 0.3941 -
acc: 0.9029 - val_loss: 0.9358 - val_acc: 0.7717
Epoch 10/15
49/49 [==============================] - 27s 549ms/step - loss: 0.3562 -
acc: 0.9100 - val_loss: 0.9314 - val_acc: 0.7601
Epoch 11/15
49/49 [==============================] - 27s 552ms/step - loss: 0.3319 -
acc: 0.9194 - val_loss: 0.9337 - val_acc: 0.7691
Epoch 12/15
49/49 [==============================] - 27s 547ms/step - loss: 0.3037 -
acc: 0.9226 - val_loss: 0.9283 - val_acc: 0.7665
Epoch 13/15
49/49 [==============================] - 27s 550ms/step - loss: 0.2698 -
acc: 0.9309 - val_loss: 0.9159 - val_acc: 0.7698
Epoch 14/15
49/49 [==============================] - 27s 546ms/step - loss: 0.2373 -
acc: 0.9425 - val_loss: 0.9318 - val_acc: 0.7646
Epoch 15/15
49/49 [==============================] - 27s 545ms/step - loss: 0.2175 -
acc: 0.9395 - val_loss: 0.9452 - val_acc: 0.7665
```

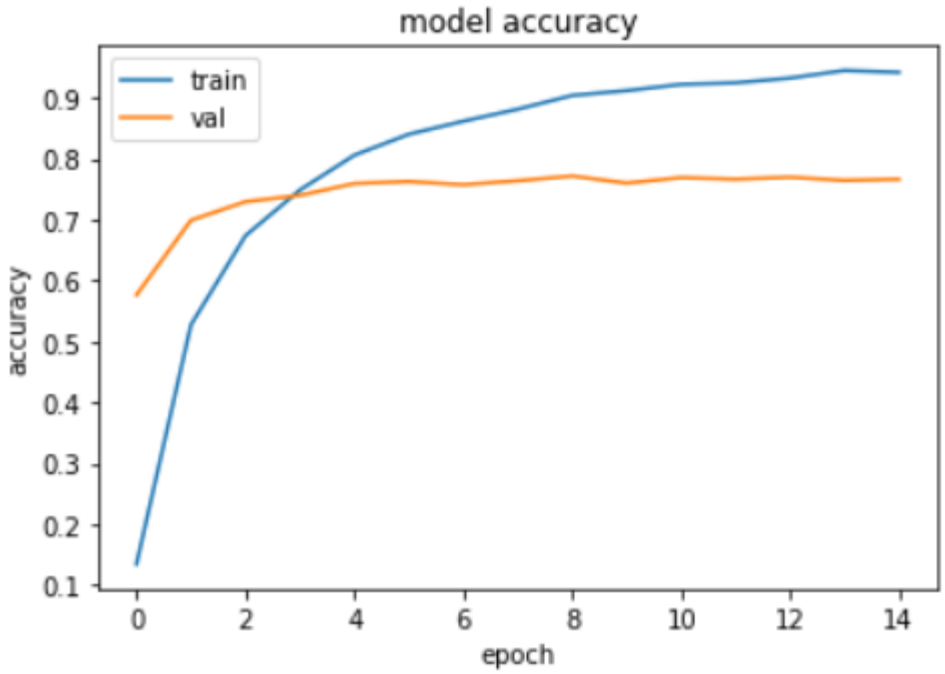
Output Run 3:

```
[17] score = model.evaluate(test_flow)

     print('The model achieved a accuracy of %.2f%%.' % (score[1]*100))
```

```
716/716 [==============================] - 98s 137ms/step
The model achieved a accuracy of 76.55%.
```

Deep Learning Project Report



<Figure size 432x288 with 0 Axes>

Result:

	First Run	Second Run	Third Run	Average
Accuracy	77.42%	76.76%	76.55%	76.91%
Training Time	402.564s	400.454s	407.144s	403.188s

Final Accuracy Result:

Accuracy from Cifar-10: 98.89%
Accuracy from Caltech-101: 92.91%
Accuracy from Caltech-256: 76.91%

Overall Accuracy = 89.57%

Deep Learning Project Report