"To iterate is human, to recurse divine."

AuxoLabs Birds Classify : *Using Transfer Learning on InceptionV3*

Download dataset from here

Dataset description

Caltech-UCSD Birds 200 (CUB-200) is an image dataset with photos of 200 bird species (mostly North American).

```
Number of categories: 200

Number of images: 6,033
```

Getting started

The goal of this project is to use transfer learning and fine-tuning to identify any bird classes from the given dataset.

We follow the below two steps in order:

- **Transfer learning**: Let us take a ConvNet that has been pre-trained on ImageNet, remove the last fully-connected layer, then treat the rest of the ConvNet as a feature extractor for the new dataset. Once we extract the features for all images, then train a classifier for the new dataset.
- **Fine-tuning**: Now replace and retrain the classifier on top of the ConvNet, and also fine-tune the weights of the pre-trained network via backpropagation.

Software to pre-install

This is supported for Python >= 2.7 and Python >= 3.3.

Dependencies :

```
Pillow==5.0.0
numpy==1.14.0
Tensorflow==1.5.0
Keras==2.1.3
```

Approach

Please follow the below steps.

Steps:

Get data

```
wget http://www.vision.caltech.edu/visipedia-data/CUB-200/images.tgz
tar -xvzf images.tar.gz
```

Images folder contains 200 categories of Birds dataset. With a total of `6033` images.

```
chaitulukka@instance-1:~/auxolabs/images$ ls
001.Black_footed_Albatross   035.Purple_Finch                 069.Rufous_Hummingbird          103.Sayornis                     137.Cliff_Swallow                  171.Myrtle_Warbler
002.Laysan_Albatross         036.Northern_Flicker             070.Green_Violetear             104.American_Pipit               138.Tree_Swallow                   172.Nashville_Warbler
003.Sooty_Albatross          037.Acadian_Flycatcher           071.Long_tailed_Jaeger          105.Whip_poor_Will               139.Scarlet_Tanager                173.Orange_crowned_Warbler
004.Groove_billed_Ani        038.Great_Crested_Flycatcher     072.Pomarine_Jaeger             106.Horned_Puffin                140.Summer_Tanager                 174.Palm_Warbler
005.Crested_Auklet           039.Least_Flycatcher             073.Blue_Jay                    107.Common_Raven                 141.Artic_Tern                     175.Pine_Warbler
006.Least_Auklet             040.Olive_sided_Flycatcher       074.Florida_Jay                 108.White_necked_Raven           142.Black_Tern                     176.Prairie_Warbler
007.Parakeet_Auklet          041.Scissor_tailed_Flycatcher    075.Green_Jay                   109.American_Redstart            143.Caspian_Tern                   177.Prothonotary_Warbler
008.Rhinoceros_Auklet        042.Vermilion_Flycatcher         076.Dark_eyed_Junco             110.Geococcyx                    144.Common_Tern                    178.Swainson_Warbler
009.Brewer_Blackbird         043.Yellow_bellied_Flycatcher    077.Tropical_Kingbird           111.Loggerhead_Shrike            145.Elegant_Tern                   179.Tennessee_Warbler
010.Red_winged_Blackbird     044.Frigatebird                  078.Gray_Kingbird               112.Great_Grey_Shrike            146.Forsters_Tern                  180.Wilson_Warbler
011.Rusty_Blackbird          045.Northern_Fulmar              079.Belted_Kingfisher           113.Baird_Sparrow                147.Least_Tern                     181.Worm_eating_Warbler
012.Yellow_headed_Blackbird  046.Gadwall                      080.Green_Kingfisher            114.Black_throated_Sparrow       148.Green_tailed_Towhee            182.Yellow_Warbler
013.Bobolink                 047.American_Goldfinch           081.Pied_Kingfisher             115.Brewer_Sparrow               149.Brown_Thrasher                 183.Northern_Waterthrush
014.Indigo_Bunting           048.European_Goldfinch           082.Ringed_Kingfisher           116.Chipping_Sparrow             150.Sage_Thrasher                  184.Louisiana_Waterthrush
015.Lazuli_Bunting           049.Boat_tailed_Grackle          083.White_breasted_Kingfisher   117.Clay_colored_Sparrow         151.Black_capped_Vireo             185.Bohemian_Waxwing
016.Painted_Bunting          050.Eared_Grebe                  084.Red_legged_Kittiwake        118.House_Sparrow                152.Blue_headed_Vireo              186.Cedar_Waxwing
017.Cardinal                 051.Horned_Grebe                 085.Horned_Lark                 119.Field_Sparrow                153.Philadelphia_Vireo             187.American_Three_toed_Woodpecker
018.Spotted_Catbird          052.Pied_billed_Grebe            086.Pacific_Loon                120.Fox_Sparrow                  154.Red_eyed_Vireo                 188.Pileated_Woodpecker
019.Gray_Catbird             053.Western_Grebe                087.Mallard                     121.Grasshopper_Sparrow          155.Warbling_Vireo                 189.Red_bellied_Woodpecker
020.Yellow_breasted_Chat     054.Blue_Grosbeak                088.Western_Meadowlark          122.Harris_Sparrow               156.White_eyed_Vireo               190.Red_cockaded_Woodpecker
021.Eastern_Towhee           055.Evening_Grosbeak             089.Hooded_Merganser            123.Henslow_Sparrow              157.Yellow_throated_Vireo          191.Red_headed_Woodpecker
022.Chuck_will_Widow         056.Pine_Grosbeak                090.Red_breasted_Merganser      124.Le_Conte_Sparrow             158.Bay_breasted_Warbler           192.Downy_Woodpecker
023.Brandt_Cormorant         057.Rose_breasted_Grosbeak       091.Mockingbird                 125.Lincoln_Sparrow              159.Black_and_white_Warbler        193.Bewick_Wren
024.Red_faced_Cormorant      058.Pigeon_Guillemot             092.Nighthawk                   126.Nelson_Sharp_tailed_Sparrow  160.Black_throated_Blue_Warbler    194.Cactus_Wren
025.Pelagic_Cormorant        059.California_Gull              093.Clark_Nutcracker            127.Savannah_Sparrow             161.Blue_winged_Warbler            195.Carolina_Wren
026.Bronzed_Cowbird          060.Glaucous_winged_Gull         094.White_breasted_Nuthatch     128.Seaside_Sparrow              162.Canada_Warbler                 196.House_Wren
027.Shiny_Cowbird            061.Heermann_Gull                095.Baltimore_Oriole            129.Song_Sparrow                 163.Cape_May_Warbler               197.Marsh_Wren
028.Brown_Creeper            062.Herring_Gull                 096.Hooded_Oriole               130.Tree_Sparrow                 164.Cerulean_Warbler               198.Rock_Wren
029.American_Crow            063.Ivory_Gull                   097.Orchard_Oriole              131.Vesper_Sparrow               165.Chestnut_sided_Warbler         199.Winter_Wren
030.Fish_Crow                064.Ring_billed_Gull             098.Scott_Oriole                132.White_crowned_Sparrow        166.Golden_winged_Warbler          200.Common_Yellowthroat
031.Black_billed_Cuckoo      065.Slaty_backed_Gull            099.Ovenbird                    133.White_throated_Sparrow       167.Hooded_Warbler
032.Mangrove_Cuckoo          066.Western_Gull                 100.Brown_Pelican               134.Cape_Glossy_Starling         168.Kentucky_Warbler
033.Yellow_billed_Cuckoo     067.Anna_Hummingbird             101.White_Pelican               135.Bank_Swallow                 169.Magnolia_Warbler
034.Gray_crowned_Rosy_Finch  068.Ruby_throated_Hummingbird    102.Western_Wood_Pewee          136.Barn_Swallow                 170.Mourning_Warbler
chaitulukka@instance-1:~/auxolabs/images$
```

Split data:

- We then split the data into **train/test** folders in the ratio of `70:30`. i.e `70%` of images in training set and `30%` of images in validation set.
- We can also divide the data into **train/valid/test** in the ratio `60:20:20` but here, in this implementation we follow the former approach i.e `70:30`

```
    import os, sys
    import shutil
    import random
    from shutil import copyfile


    source_dir = 'images/'
    for root, dirs, files in os.walk(source_dir):
        for i in dirs:

            path = 'test/' + "%s/" % i
            os.makedirs(path)

            filenames = random.sample(os.listdir('images/' + "%s/" % i ),
```

```
int(len(os.listdir('images/' + "%s/" % i ))*0.3))
            for j in filenames:
                shutil.move('images/' + "%s/" % i  + j, path)
```

- We use the above simple script to recursively copy and randomly move the `30%` of the images of each class into our **test** directory. `filenames = random.sample(os.listdir('images/' + "%s/" % i ), int(len(os.listdir('images/' + "%s/" % i ))*0.3))`
- This above line helps use in randomly selecting `30%` of the images in each folder i.e `from 200 classes`. You can find above script in `split.py`
    - Use `find train -type f | wc -l` and `find test -type f | wc -l` to count the no. of files in train and test folders and verify the splitting process. We get `4318` and `1715` respectively.

Imports:

Call all necessary libraries and system funcs:

```
import os
import sys
import glob
import argparse
import matplotlib.pyplot as plt

from keras import __version__
from keras.applications.inception_v3 import InceptionV3, preprocess_input
from keras.models import Model
from keras.layers import Dense, GlobalAveragePooling2D
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import SGD
from keras.utils import plot_model
```

Declare all global variables:

```
IM_WIDTH, IM_HEIGHT = 299, 299
NB_EPOCHS = 3
BAT_SIZE = 32
FC_SIZE = 1024
NB_IV3_LAYERS_TO_FREEZE = 172

train_dir= 'train' # locate training folder
val_dir='test'     # locate test folder
nb_epoch=NB_EPOCHS
batch_size=BAT_SIZE
```

Data Augmentation:

- Data augmentation is the process of artificially increasing the size of your dataset via transformations.

```
# data augementation
  train_datagen =  ImageDataGenerator(
      preprocessing_function=preprocess_input,
      rotation_range=30,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True
  )
  test_datagen = ImageDataGenerator(
      preprocessing_function=preprocess_input,
      rotation_range=30,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True
  )

  train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(IM_WIDTH, IM_HEIGHT),
    batch_size=batch_size,
  )

  validation_generator = test_datagen.flow_from_directory(
    val_dir,
    target_size=(IM_WIDTH, IM_HEIGHT),
    batch_size=batch_size,
  )
```

Define base model:

```
base_model = InceptionV3(weights='imagenet', include_top=False)
#include_top=False excludes final FC layer

model = add_new_last_layer(base_model, nb_classes)
```

Call to Transfer learning func:

- Here we are using `rmsprop` optimizer we can also try `adam` and check the results accuracy.

```
def setup_to_transfer_learn(model, base_model):

  for layer in base_model.layers:
    layer.trainable = False
  model.summary()
  plot_model(model, to_file='model.png')
  model.compile(optimizer='rmsprop', loss='categorical_crossentropy',
metrics=['accuracy'])
```

- `model.summary()` helps us in analyzing the model and check the parameters in each layer.

> **Note**: All layers are not displayed below. Just for intuition purpose only.

```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, None, None, 3 0

conv2d_1 (Conv2D)               (None, None, None, 3 864         input_1[0][0]

batch_normalization_1 (BatchNor (None, None, None, 3 96          conv2d_1[0][0]

activation_1 (Activation)       (None, None, None, 3 0           batch_normalization_1[0][0]

conv2d_2 (Conv2D)               (None, None, None, 3 9216        activation_1[0][0]

batch_normalization_2 (BatchNor (None, None, None, 3 96          conv2d_2[0][0]
.
.
.
.
.
.
.
global_average_pooling2d_1 (Glo (None, 2048)         0           mixed10[0][0]

dense_1 (Dense)                 (None, 1024)         2098176     global_average_pooling2d_1[0][0]

dense_2 (Dense)                 (None, 200)          205000      dense_1[0][0]
==================================================================================================
```

Call to fine-tuning func:

> Here we are using standard learning rate and momentum parameters. i.e `lr=0.0001` and `momentum=0.9`

```
def setup_to_finetune(model):
  """Freeze the bottom 172 and retrain the remaining top layers.

  for layer in model.layers[:172]:
     layer.trainable = False
```

```
  for layer in model.layers[172:]:
     layer.trainable = True
  model.compile(optimizer=SGD(lr=0.0001, momentum=0.9),
loss='categorical_crossentropy', metrics=['accuracy'])
```

Fit Model: using model.fit_generator()

> Fits the model on data generated batch-by-batch by a Python generator. The generator is run in parallel to the model, for efficiency.

For Transfer Learning:

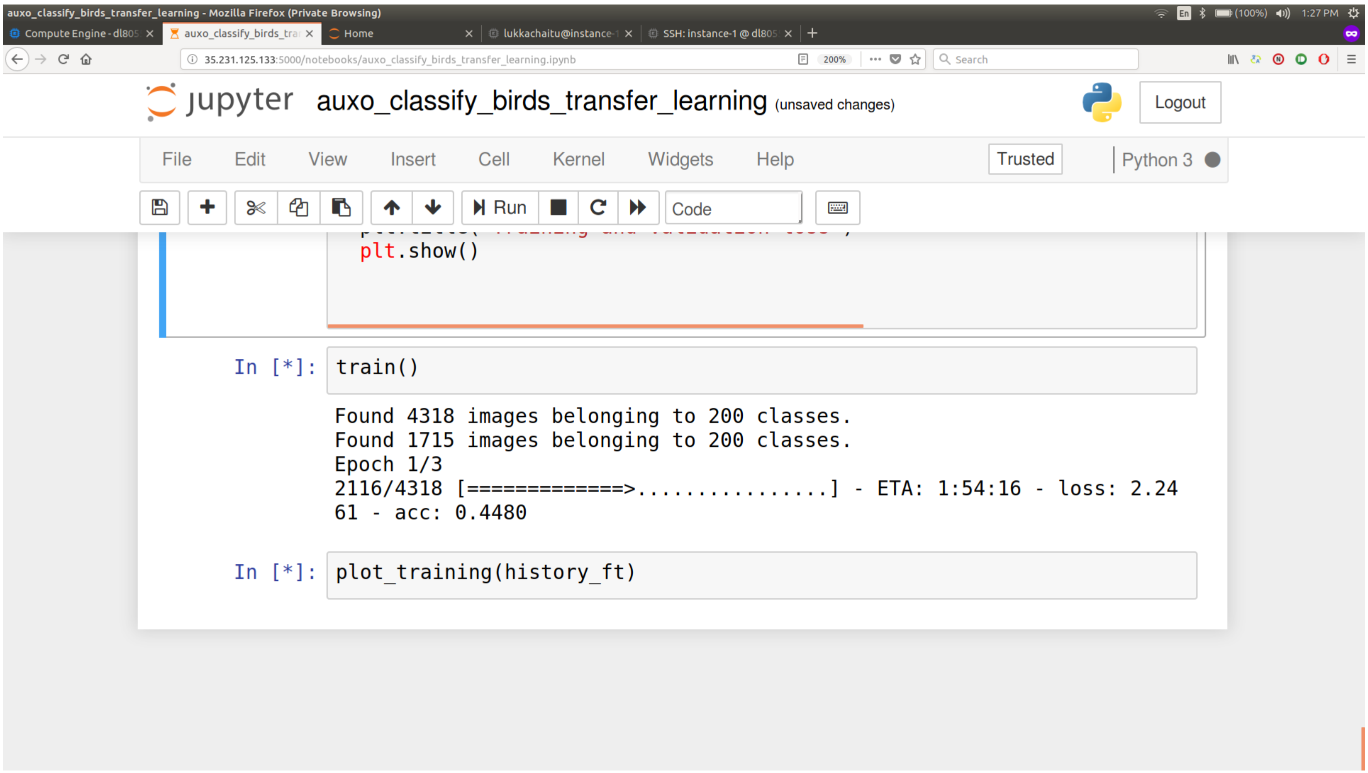
```
history_tl = model.fit_generator(
  train_generator,
  epochs=nb_epoch,
  steps_per_epoch=nb_train_samples,
  validation_data=validation_generator,
  validation_steps=nb_val_samples,
  class_weight='auto')
```

Then for Fine-tuning:

```
history_ft = model.fit_generator(
  train_generator,
  steps_per_epoch=nb_train_samples,
  epochs=nb_epoch,
  validation_data=validation_generator,
  validation_steps=nb_val_samples,
  class_weight='auto')
```

Training:

Call `train()` to start the training process. Once the process is started we can see an output as below:



Finally *Save* the model:

Do not forget to save the model, you can use: `model.save('lc_auxo_birds.h5')`

Plotting the Cost vs Epochs:

Use the `plot_training()` func to plot the

```
def plot_training(history):
  acc = history.history['acc']
  val_acc = history.history['val_acc']
  loss = history.history['loss']
  val_loss = history.history['val_loss']
  epochs = range(len(acc))

  plt.plot(epochs, acc, 'r.')
  plt.plot(epochs, val_acc, 'r')
  plt.title('Training and validation accuracy')

  plt.figure()
  plt.plot(epochs, loss, 'r.')
  plt.plot(epochs, val_loss, 'r-')
  plt.title('Training and validation loss')
  plt.show()
```

Using the saved model:

```
import keras
from keras.models import load_model
from keras.models import Sequential
import cv2
import numpy as np
from keras.preprocessing.image import ImageDataGenerator, array_to_img,
img_to_array, load_img
model = Sequential()

model =load_model('lc_auxo_birds.h5') ##load the saved model in the previous step
model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])

img = cv2.imread('images/018.Spotted_Catbird/Spotted_Catbird_0015_3026208002.jpg')
img = cv2.resize(img,(299,299))
img = np.reshape(img,[1,299,299,3])
classes = model.predict_classes(img)
print classes
```

`model.predict_classes()` outputs the predicted image category.

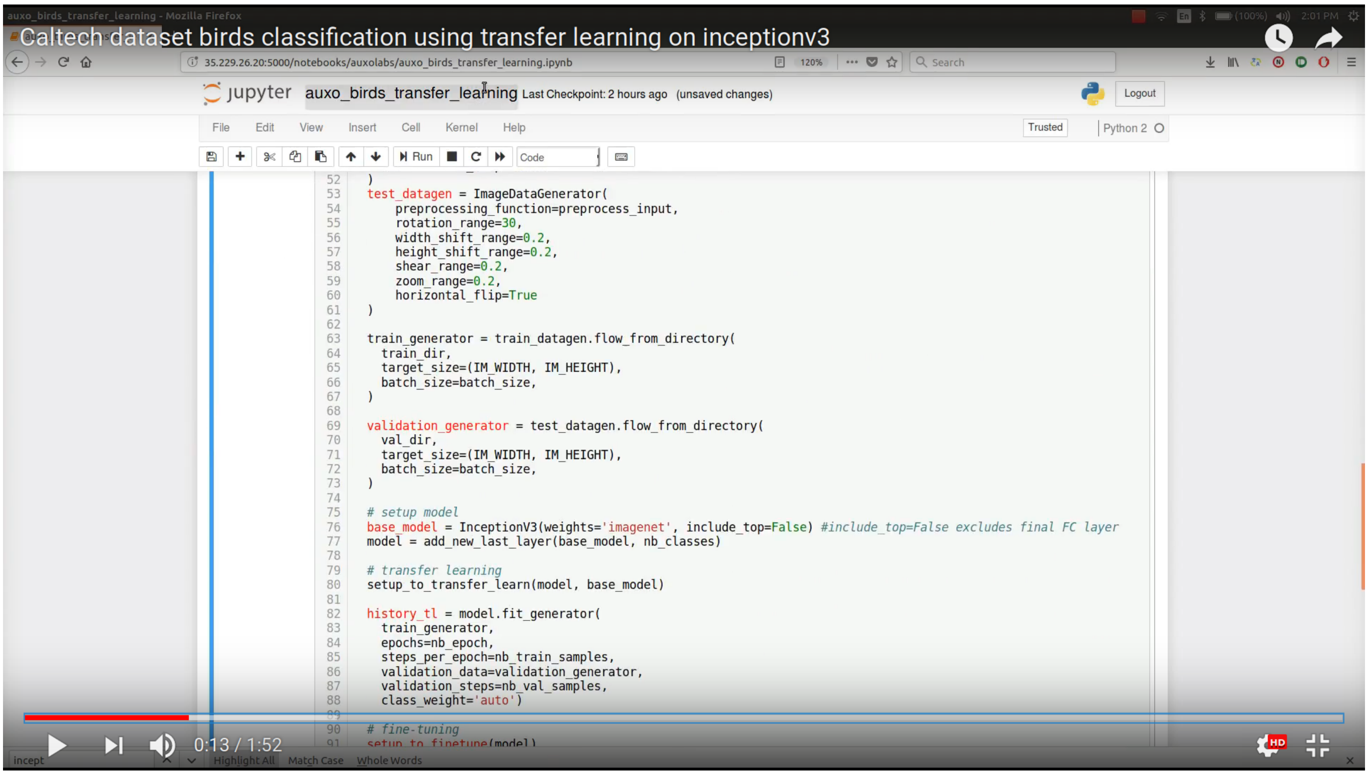
Command line usage:

```
sudo python3 auxo_tl.py --train_dir train --val_dir test
```

Future scope to improve performance:

- We may use L2 regularizer to improve the performance.
- Can also add Batch normalization layer to avoid the overfitting.
- We use gradient checking to evaluate the back propagation.

Screencast:



License

[MIT License]