dog_app

April 6, 2020

1 Convolutional Neural Networks

1.1 Project: Write an Algorithm for a Dog Identification App

In this notebook, some template code has already been provided for you, and you will need to implement additional functionality to successfully complete this project. You will not need to modify the included code beyond what is requested. Sections that begin with **'(IMPLEMENTATION)'** in the header indicate that the following block of code will require additional functionality which you must provide. Instructions will be provided for each section, and the specifics of the implementation are marked in the code block with a 'TODO' statement. Please be sure to read the instructions carefully!

> **Note**: Once you have completed all of the code implementations, you need to finalize your work by exporting the Jupyter Notebook as an HTML document. Before exporting the notebook to html, all of the code cells need to have been run so that reviewers can see the final implementation and output. You can then export the notebook by using the menu above and navigating to **File -> Download as -> HTML (.html)**. Include the finished document along with this notebook as your submission.

In addition to implementing code, there will be questions that you must answer which relate to the project and your implementation. Each section where you will answer a question is preceded by a **'Question X'** header. Carefully read each question and provide thorough answers in the following text boxes that begin with **'Answer:'**. Your project submission will be evaluated based on your answers to each of the questions and the implementation you provide.

> **Note:** Code and Markdown cells can be executed using the **Shift + Enter** keyboard shortcut. Markdown cells can be edited by double-clicking the cell to enter edit mode.

The rubric contains *optional* "Stand Out Suggestions" for enhancing the project beyond the minimum requirements. If you decide to pursue the "Stand Out Suggestions", you should include the code in this Jupyter notebook.

Step 0: Import Datasets

Make sure that you've downloaded the required human and dog datasets: * Download the dog dataset. Unzip the folder and place it in this project's home directory, at the location `/dogImages`.

- Download the human dataset. Unzip the folder and place it in the home directory, at location `/lfw`.

Note: If you are using a Windows machine, you are encouraged to use 7zip to extract the folder.

In the code cell below, we save the file paths for both the human (LFW) dataset and dog dataset in the numpy arrays `human_files` and `dog_files`.

[0]:
```
'''
import requests

human_download = 'https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/
 ↪lfw.zip'
dog_download = 'https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/
 ↪dogImages.zip'

downloads = [('human', human_download), ('dog', dog_download)]

for download in downloads:
  r = requests.get(download[1], stream = True)

  with open(download[0] + ".zip", "wb") as file:
      for block in r.iter_content(chunk_size = 1024):
          if block:
              file.write(block)


!unzip -uq "dog.zip" -d "dog_folder"
!unzip -uq "human.zip" -d "/dog_folder"
'''
```

[0]:
```
'\nimport requests\n\nhuman_download = \'https://s3-us-west-1.amazonaws.com
/udacity-aind/dog-project/lfw.zip\'\ndog_download = \'https://s3-us-
west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip\'\n\ndownloads =
[(\'human\', human_download), (\'dog\', dog_download)]\n\nfor download in
downloads:\n  r = requests.get(download[1], stream = True)  \n\n  with
open(download[0] + ".zip", "wb") as file:  \n      for block in
r.iter_content(chunk_size = 1024): \n          if block:  \n
file.write(block) \n\n\n!unzip -uq "dog.zip" -d "dog_folder"\n!unzip -uq
"human.zip" -d "/dog_folder"\n'
```

[1]:
```
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!pip install pypandoc
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
pandoc is already the newest version (1.19.2.4~dfsg-1build4).
pandoc set to manually installed.
The following additional packages will be installed:
  fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-texgyre
  javascript-common libcupsfilters1 libcupsimage2 libgs9 libgs9-common
  libijs-0.35 libjbig2dec0 libjs-jquery libkpathsea6 libpotrace0 libptexenc1
```

```
  libruby2.5 libsynctex1 libtexlua52 libtexluajit2 libzzip-0-13 lmodern
  poppler-data preview-latex-style rake ruby ruby-did-you-mean ruby-minitest
  ruby-net-telnet ruby-power-assert ruby-test-unit ruby2.5
  rubygems-integration t1utils tex-common tex-gyre texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base
  texlive-latex-recommended texlive-pictures texlive-plain-generic tipa
Suggested packages:
  fonts-noto apache2 | lighttpd | httpd poppler-utils ghostscript
  fonts-japanese-mincho | fonts-ipafont-mincho fonts-japanese-gothic
  | fonts-ipafont-gothic fonts-arphic-ukai fonts-arphic-uming fonts-nanum ri
  ruby-dev bundler debhelper gv | postscript-viewer perl-tk xpdf-reader
  | pdf-viewer texlive-fonts-recommended-doc texlive-latex-base-doc
  python-pygments icc-profiles libfile-which-perl
  libspreadsheet-parseexcel-perl texlive-latex-extra-doc
  texlive-latex-recommended-doc texlive-pstricks dot2tex prerex ruby-tcltk
  | libtcltk-ruby texlive-pictures-doc vprerex
The following NEW packages will be installed:
  fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-texgyre
  javascript-common libcupsfilters1 libcupsimage2 libgs9 libgs9-common
  libijs-0.35 libjbig2dec0 libjs-jquery libkpathsea6 libpotrace0 libptexenc1
  libruby2.5 libsynctex1 libtexlua52 libtexluajit2 libzzip-0-13 lmodern
  poppler-data preview-latex-style rake ruby ruby-did-you-mean ruby-minitest
  ruby-net-telnet ruby-power-assert ruby-test-unit ruby2.5
  rubygems-integration t1utils tex-common tex-gyre texlive texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base
  texlive-latex-extra texlive-latex-recommended texlive-pictures
  texlive-plain-generic texlive-xetex tipa
0 upgraded, 47 newly installed, 0 to remove and 25 not upgraded.
Need to get 146 MB of archives.
After this operation, 460 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-droid-fallback
all 1:6.0.1r16-1.1 [1,805 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-lato all 2.0-2
[2,698 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic/main amd64 poppler-data all
0.4.8-2 [1,479 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic/main amd64 tex-common all 6.09
[33.0 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-lmodern all
2.004.5-3 [4,551 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-noto-mono all
20171026-2 [75.5 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic/universe amd64 fonts-texgyre all
20160520-1 [8,761 kB]
Get:8 http://archive.ubuntu.com/ubuntu bionic/main amd64 javascript-common all
11 [6,066 B]
Get:9 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcupsfilters1
amd64 1.20.2-0ubuntu3.1 [108 kB]
```

```
Get:10 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcupsimage2
amd64 2.2.7-1ubuntu2.7 [18.6 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic/main amd64 libijs-0.35 amd64
0.35-13 [15.5 kB]
Get:12 http://archive.ubuntu.com/ubuntu bionic/main amd64 libjbig2dec0 amd64
0.13-6 [55.9 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgs9-common
all 9.26~dfsg+0-0ubuntu0.18.04.12 [5,092 kB]
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgs9 amd64
9.26~dfsg+0-0ubuntu0.18.04.12 [2,264 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic/main amd64 libjs-jquery all
3.2.1-1 [152 kB]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libkpathsea6
amd64 2017.20170613.44572-8ubuntu0.1 [54.9 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic/main amd64 libpotrace0 amd64
1.14-2 [17.4 kB]
Get:18 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libptexenc1
amd64 2017.20170613.44572-8ubuntu0.1 [34.5 kB]
Get:19 http://archive.ubuntu.com/ubuntu bionic/main amd64 rubygems-integration
all 1.11 [4,994 B]
Get:20 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 ruby2.5 amd64
2.5.1-1ubuntu1.6 [48.6 kB]
Get:21 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby amd64 1:2.5.1
[5,712 B]
Get:22 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 rake all
12.3.1-1ubuntu0.1 [44.9 kB]
Get:23 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-did-you-mean all
1.2.0-2 [9,700 B]
Get:24 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-minitest all
5.10.3-1 [38.6 kB]
Get:25 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-net-telnet all
0.1.1-2 [12.6 kB]
Get:26 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-power-assert all
0.3.0-1 [7,952 B]
Get:27 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-test-unit all
3.2.5-1 [61.1 kB]
Get:28 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libruby2.5
amd64 2.5.1-1ubuntu1.6 [3,069 kB]
Get:29 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libsynctex1
amd64 2017.20170613.44572-8ubuntu0.1 [41.4 kB]
Get:30 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libtexlua52
amd64 2017.20170613.44572-8ubuntu0.1 [91.2 kB]
Get:31 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libtexluajit2
amd64 2017.20170613.44572-8ubuntu0.1 [230 kB]
Get:32 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libzzip-0-13
amd64 0.13.62-3.1ubuntu0.18.04.1 [26.0 kB]
Get:33 http://archive.ubuntu.com/ubuntu bionic/main amd64 lmodern all 2.004.5-3
[9,631 kB]
```

```
Get:34 http://archive.ubuntu.com/ubuntu bionic/main amd64 preview-latex-style
all 11.91-1ubuntu1 [185 kB]
Get:35 http://archive.ubuntu.com/ubuntu bionic/main amd64 t1utils amd64 1.41-2
[56.0 kB]
Get:36 http://archive.ubuntu.com/ubuntu bionic/universe amd64 tex-gyre all
20160520-1 [4,998 kB]
Get:37 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 texlive-
binaries amd64 2017.20170613.44572-8ubuntu0.1 [8,179 kB]
Get:38 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-base all
2017.20180305-1 [18.7 MB]
Get:39 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-fonts-
recommended all 2017.20180305-1 [5,262 kB]
Get:40 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-latex-base all
2017.20180305-1 [951 kB]
Get:41 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-latex-
recommended all 2017.20180305-1 [14.9 MB]
Get:42 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive all
2017.20180305-1 [14.4 kB]
Get:43 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-pictures
all 2017.20180305-1 [4,026 kB]
Get:44 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-latex-
extra all 2017.20180305-2 [10.6 MB]
Get:45 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-plain-
generic all 2017.20180305-2 [23.6 MB]
Get:46 http://archive.ubuntu.com/ubuntu bionic/universe amd64 tipa all 2:1.3-20
[2,978 kB]
Get:47 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-xetex all
2017.20180305-1 [10.7 MB]
Fetched 146 MB in 19s (7,847 kB/s)
Extracting templates from packages: 100%
Preconfiguring packages ...
Selecting previously unselected package fonts-droid-fallback.
(Reading database ... 144542 files and directories currently installed.)
Preparing to unpack .../00-fonts-droid-fallback_1%3a6.0.1r16-1.1_all.deb ...
Unpacking fonts-droid-fallback (1:6.0.1r16-1.1) ...
Selecting previously unselected package fonts-lato.
Preparing to unpack .../01-fonts-lato_2.0-2_all.deb ...
Unpacking fonts-lato (2.0-2) ...
Selecting previously unselected package poppler-data.
Preparing to unpack .../02-poppler-data_0.4.8-2_all.deb ...
Unpacking poppler-data (0.4.8-2) ...
Selecting previously unselected package tex-common.
Preparing to unpack .../03-tex-common_6.09_all.deb ...
Unpacking tex-common (6.09) ...
Selecting previously unselected package fonts-lmodern.
Preparing to unpack .../04-fonts-lmodern_2.004.5-3_all.deb ...
Unpacking fonts-lmodern (2.004.5-3) ...
Selecting previously unselected package fonts-noto-mono.
```

```
Preparing to unpack .../05-fonts-noto-mono_20171026-2_all.deb ...
Unpacking fonts-noto-mono (20171026-2) ...
Selecting previously unselected package fonts-texgyre.
Preparing to unpack .../06-fonts-texgyre_20160520-1_all.deb ...
Unpacking fonts-texgyre (20160520-1) ...
Selecting previously unselected package javascript-common.
Preparing to unpack .../07-javascript-common_11_all.deb ...
Unpacking javascript-common (11) ...
Selecting previously unselected package libcupsfilters1:amd64.
Preparing to unpack .../08-libcupsfilters1_1.20.2-0ubuntu3.1_amd64.deb ...
Unpacking libcupsfilters1:amd64 (1.20.2-0ubuntu3.1) ...
Selecting previously unselected package libcupsimage2:amd64.
Preparing to unpack .../09-libcupsimage2_2.2.7-1ubuntu2.7_amd64.deb ...
Unpacking libcupsimage2:amd64 (2.2.7-1ubuntu2.7) ...
Selecting previously unselected package libijs-0.35:amd64.
Preparing to unpack .../10-libijs-0.35_0.35-13_amd64.deb ...
Unpacking libijs-0.35:amd64 (0.35-13) ...
Selecting previously unselected package libjbig2dec0:amd64.
Preparing to unpack .../11-libjbig2dec0_0.13-6_amd64.deb ...
Unpacking libjbig2dec0:amd64 (0.13-6) ...
Selecting previously unselected package libgs9-common.
Preparing to unpack .../12-libgs9-common_9.26~dfsg+0-0ubuntu0.18.04.12_all.deb
...
Unpacking libgs9-common (9.26~dfsg+0-0ubuntu0.18.04.12) ...
Selecting previously unselected package libgs9:amd64.
Preparing to unpack .../13-libgs9_9.26~dfsg+0-0ubuntu0.18.04.12_amd64.deb ...
Unpacking libgs9:amd64 (9.26~dfsg+0-0ubuntu0.18.04.12) ...
Selecting previously unselected package libjs-jquery.
Preparing to unpack .../14-libjs-jquery_3.2.1-1_all.deb ...
Unpacking libjs-jquery (3.2.1-1) ...
Selecting previously unselected package libkpathsea6:amd64.
Preparing to unpack .../15-libkpathsea6_2017.20170613.44572-8ubuntu0.1_amd64.deb
...
Unpacking libkpathsea6:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Selecting previously unselected package libpotrace0.
Preparing to unpack .../16-libpotrace0_1.14-2_amd64.deb ...
Unpacking libpotrace0 (1.14-2) ...
Selecting previously unselected package libptexenc1:amd64.
Preparing to unpack .../17-libptexenc1_2017.20170613.44572-8ubuntu0.1_amd64.deb
...
Unpacking libptexenc1:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Selecting previously unselected package rubygems-integration.
Preparing to unpack .../18-rubygems-integration_1.11_all.deb ...
Unpacking rubygems-integration (1.11) ...
Selecting previously unselected package ruby2.5.
Preparing to unpack .../19-ruby2.5_2.5.1-1ubuntu1.6_amd64.deb ...
Unpacking ruby2.5 (2.5.1-1ubuntu1.6) ...
Selecting previously unselected package ruby.
```

```
Preparing to unpack .../20-ruby_1%3a2.5.1_amd64.deb ...
Unpacking ruby (1:2.5.1) ...
Selecting previously unselected package rake.
Preparing to unpack .../21-rake_12.3.1-1ubuntu0.1_all.deb ...
Unpacking rake (12.3.1-1ubuntu0.1) ...
Selecting previously unselected package ruby-did-you-mean.
Preparing to unpack .../22-ruby-did-you-mean_1.2.0-2_all.deb ...
Unpacking ruby-did-you-mean (1.2.0-2) ...
Selecting previously unselected package ruby-minitest.
Preparing to unpack .../23-ruby-minitest_5.10.3-1_all.deb ...
Unpacking ruby-minitest (5.10.3-1) ...
Selecting previously unselected package ruby-net-telnet.
Preparing to unpack .../24-ruby-net-telnet_0.1.1-2_all.deb ...
Unpacking ruby-net-telnet (0.1.1-2) ...
Selecting previously unselected package ruby-power-assert.
Preparing to unpack .../25-ruby-power-assert_0.3.0-1_all.deb ...
Unpacking ruby-power-assert (0.3.0-1) ...
Selecting previously unselected package ruby-test-unit.
Preparing to unpack .../26-ruby-test-unit_3.2.5-1_all.deb ...
Unpacking ruby-test-unit (3.2.5-1) ...
Selecting previously unselected package libruby2.5:amd64.
Preparing to unpack .../27-libruby2.5_2.5.1-1ubuntu1.6_amd64.deb ...
Unpacking libruby2.5:amd64 (2.5.1-1ubuntu1.6) ...
Selecting previously unselected package libsynctex1:amd64.
Preparing to unpack .../28-libsynctex1_2017.20170613.44572-8ubuntu0.1_amd64.deb
...
Unpacking libsynctex1:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Selecting previously unselected package libtexlua52:amd64.
Preparing to unpack .../29-libtexlua52_2017.20170613.44572-8ubuntu0.1_amd64.deb
...
Unpacking libtexlua52:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Selecting previously unselected package libtexluajit2:amd64.
Preparing to unpack
.../30-libtexluajit2_2017.20170613.44572-8ubuntu0.1_amd64.deb ...
Unpacking libtexluajit2:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Selecting previously unselected package libzzip-0-13:amd64.
Preparing to unpack .../31-libzzip-0-13_0.13.62-3.1ubuntu0.18.04.1_amd64.deb ...
Unpacking libzzip-0-13:amd64 (0.13.62-3.1ubuntu0.18.04.1) ...
Selecting previously unselected package lmodern.
Preparing to unpack .../32-lmodern_2.004.5-3_all.deb ...
Unpacking lmodern (2.004.5-3) ...
Selecting previously unselected package preview-latex-style.
Preparing to unpack .../33-preview-latex-style_11.91-1ubuntu1_all.deb ...
Unpacking preview-latex-style (11.91-1ubuntu1) ...
Selecting previously unselected package t1utils.
Preparing to unpack .../34-t1utils_1.41-2_amd64.deb ...
Unpacking t1utils (1.41-2) ...
Selecting previously unselected package tex-gyre.
```

```
Preparing to unpack .../35-tex-gyre_20160520-1_all.deb ...
Unpacking tex-gyre (20160520-1) ...
Selecting previously unselected package texlive-binaries.
Preparing to unpack .../36-texlive-
binaries_2017.20170613.44572-8ubuntu0.1_amd64.deb ...
Unpacking texlive-binaries (2017.20170613.44572-8ubuntu0.1) ...
Selecting previously unselected package texlive-base.
Preparing to unpack .../37-texlive-base_2017.20180305-1_all.deb ...
Unpacking texlive-base (2017.20180305-1) ...
Selecting previously unselected package texlive-fonts-recommended.
Preparing to unpack .../38-texlive-fonts-recommended_2017.20180305-1_all.deb ...
Unpacking texlive-fonts-recommended (2017.20180305-1) ...
Selecting previously unselected package texlive-latex-base.
Preparing to unpack .../39-texlive-latex-base_2017.20180305-1_all.deb ...
Unpacking texlive-latex-base (2017.20180305-1) ...
Selecting previously unselected package texlive-latex-recommended.
Preparing to unpack .../40-texlive-latex-recommended_2017.20180305-1_all.deb ...
Unpacking texlive-latex-recommended (2017.20180305-1) ...
Selecting previously unselected package texlive.
Preparing to unpack .../41-texlive_2017.20180305-1_all.deb ...
Unpacking texlive (2017.20180305-1) ...
Selecting previously unselected package texlive-pictures.
Preparing to unpack .../42-texlive-pictures_2017.20180305-1_all.deb ...
Unpacking texlive-pictures (2017.20180305-1) ...
Selecting previously unselected package texlive-latex-extra.
Preparing to unpack .../43-texlive-latex-extra_2017.20180305-2_all.deb ...
Unpacking texlive-latex-extra (2017.20180305-2) ...
Selecting previously unselected package texlive-plain-generic.
Preparing to unpack .../44-texlive-plain-generic_2017.20180305-2_all.deb ...
Unpacking texlive-plain-generic (2017.20180305-2) ...
Selecting previously unselected package tipa.
Preparing to unpack .../45-tipa_2%3a1.3-20_all.deb ...
Unpacking tipa (2:1.3-20) ...
Selecting previously unselected package texlive-xetex.
Preparing to unpack .../46-texlive-xetex_2017.20180305-1_all.deb ...
Unpacking texlive-xetex (2017.20180305-1) ...
Setting up libgs9-common (9.26~dfsg+0-0ubuntu0.18.04.12) ...
Setting up libkpathsea6:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Setting up libjs-jquery (3.2.1-1) ...
Setting up libtexlua52:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Setting up fonts-droid-fallback (1:6.0.1r16-1.1) ...
Setting up libsynctex1:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Setting up libptexenc1:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Setting up tex-common (6.09) ...
update-language: texlive-base not installed and configured, doing nothing!
Setting up poppler-data (0.4.8-2) ...
Setting up tex-gyre (20160520-1) ...
Setting up preview-latex-style (11.91-1ubuntu1) ...
```

```
Setting up fonts-texgyre (20160520-1) ...
Setting up fonts-noto-mono (20171026-2) ...
Setting up fonts-lato (2.0-2) ...
Setting up libcupsfilters1:amd64 (1.20.2-0ubuntu3.1) ...
Setting up libcupsimage2:amd64 (2.2.7-1ubuntu2.7) ...
Setting up libjbig2dec0:amd64 (0.13-6) ...
Setting up ruby-did-you-mean (1.2.0-2) ...
Setting up t1utils (1.41-2) ...
Setting up ruby-net-telnet (0.1.1-2) ...
Setting up libijs-0.35:amd64 (0.35-13) ...
Setting up rubygems-integration (1.11) ...
Setting up libpotrace0 (1.14-2) ...
Setting up javascript-common (11) ...
Setting up ruby-minitest (5.10.3-1) ...
Setting up libzzip-0-13:amd64 (0.13.62-3.1ubuntu0.18.04.1) ...
Setting up libgs9:amd64 (9.26~dfsg+0-0ubuntu0.18.04.12) ...
Setting up libtexluajit2:amd64 (2017.20170613.44572-8ubuntu0.1) ...
Setting up fonts-lmodern (2.004.5-3) ...
Setting up ruby-power-assert (0.3.0-1) ...
Setting up texlive-binaries (2017.20170613.44572-8ubuntu0.1) ...
update-alternatives: using /usr/bin/xdvi-xaw to provide /usr/bin/xdvi.bin
(xdvi.bin) in auto mode
update-alternatives: using /usr/bin/bibtex.original to provide /usr/bin/bibtex
(bibtex) in auto mode
Setting up texlive-base (2017.20180305-1) ...
mktexlsr: Updating /var/lib/texmf/ls-R-TEXLIVEDIST...
mktexlsr: Updating /var/lib/texmf/ls-R-TEXMFMAIN...
mktexlsr: Updating /var/lib/texmf/ls-R...
mktexlsr: Done.
tl-paper: setting paper size for dvips to a4: /var/lib/texmf/dvips/config
/config-paper.ps
tl-paper: setting paper size for dvipdfmx to a4: /var/lib/texmf/dvipdfmx
/dvipdfmx-paper.cfg
tl-paper: setting paper size for xdvi to a4: /var/lib/texmf/xdvi/XDvi-paper
tl-paper: setting paper size for pdftex to a4:
/var/lib/texmf/tex/generic/config/pdftexconfig.tex
Setting up texlive-fonts-recommended (2017.20180305-1) ...
Setting up texlive-plain-generic (2017.20180305-2) ...
Setting up texlive-latex-base (2017.20180305-1) ...
Setting up lmodern (2.004.5-3) ...
Setting up texlive-latex-recommended (2017.20180305-1) ...
Setting up texlive-pictures (2017.20180305-1) ...
Setting up tipa (2:1.3-20) ...
Regenerating '/var/lib/texmf/fmtutil.cnf-DEBIAN'... done.
Regenerating '/var/lib/texmf/fmtutil.cnf-TEXLIVEDIST'... done.
update-fmtutil has updated the following file(s):
        /var/lib/texmf/fmtutil.cnf-DEBIAN
        /var/lib/texmf/fmtutil.cnf-TEXLIVEDIST
```

```
If you want to activate the changes in the above file(s),
you should run fmtutil-sys or fmtutil.
Setting up texlive (2017.20180305-1) ...
Setting up texlive-latex-extra (2017.20180305-2) ...
Setting up texlive-xetex (2017.20180305-1) ...
Setting up ruby2.5 (2.5.1-1ubuntu1.6) ...
Setting up ruby (1:2.5.1) ...
Setting up ruby-test-unit (3.2.5-1) ...
Setting up rake (12.3.1-1ubuntu0.1) ...
Setting up libruby2.5:amd64 (2.5.1-1ubuntu1.6) ...
Processing triggers for mime-support (3.60ubuntu1) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
/sbin/ldconfig.real: /usr/local/lib/python3.6/dist-
packages/ideep4py/lib/libmkldnn.so.0 is not a symbolic link

Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for fontconfig (2.12.6-0ubuntu2) ...
Processing triggers for tex-common (6.09) ...
Running updmap-sys. This may take some time... done.
Running mktexlsr /var/lib/texmf ... done.
Building format(s) --all.
        This may take some time... done.
Collecting pypandoc
  Downloading https://files.pythonhosted.org/packages/71/81/00184643e5a10a456b41
18fc12c96780823adb8ed974eb2289f29703b29b/pypandoc-1.4.tar.gz
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-
packages (from pypandoc) (46.0.0)
Requirement already satisfied: pip>=8.1.0 in /usr/local/lib/python3.6/dist-
packages (from pypandoc) (19.3.1)
Requirement already satisfied: wheel>=0.25.0 in /usr/local/lib/python3.6/dist-
packages (from pypandoc) (0.34.2)
Building wheels for collected packages: pypandoc
  Building wheel for pypandoc (setup.py) ... done
  Created wheel for pypandoc: filename=pypandoc-1.4-cp36-none-any.whl size=16718
sha256=49c5f346e9c1d47811c798040888669003a42e6e06f97521e45fc4ae5e68bd7f
  Stored in directory: /root/.cache/pip/wheels/3e/55/4f/59e0fa0914f3db52e87c0642
c5fb986871dfbbf253026e639f
Successfully built pypandoc
Installing collected packages: pypandoc
Successfully installed pypandoc-1.4
```

[2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
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&response_type=code&scope=email%20http
```

```
s%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.c
om%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.reado
nly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
ůůůůůůůůůů
Mounted at /content/drive
```

[4]:
```
!cp 'drive/My Drive/Colab Notebooks/dog_app.ipynb' ./
```

```
cp: cannot stat 'drive/My Drive/Colab Notebooks/dog_app.ipynb': No such file or
directory
```

[0]:
```
import numpy as np
from glob import glob
from PIL import Image


files_path = '/dog_folder'

# load filenames for human and dog images
human_files = np.array(glob(files_path + "/lfw/*/*"))
dog_files = np.array(glob(files_path + "/dogImages/*/*/*"))

# print number of images in each dataset
print('There are %d total human images.' % len(human_files))
print('There are %d total dog images.' % len(dog_files))
```

```
There are 13233 total human images.
There are 8351 total dog images.
```

Step 1: Detect Humans

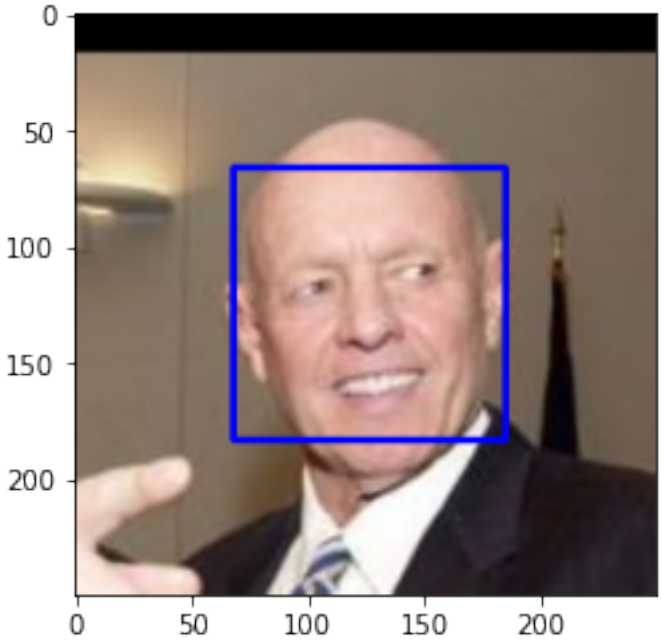
In this section, we use OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images.

OpenCV provides many pre-trained face detectors, stored as XML files on github. We have downloaded one of these detectors and stored it in the `haarcascades` directory. In the next code cell, we demonstrate how to use this detector to find human faces in a sample image.

[0]:
```
import cv2
import matplotlib.pyplot as plt
%matplotlib inline

haar_path = '/dog_human'

# extract pre-trained face detector
face_cascade = cv2.CascadeClassifier(haar_path + '/haarcascades/
 ↪haarcascade_frontalface_alt.xml')
```

```
# load color (BGR) image
img = cv2.imread(human_files[1])
# convert BGR image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# find faces in image
faces = face_cascade.detectMultiScale(gray)

# print number of faces detected in the image
print('Number of faces detected:', len(faces))

# get bounding box for each detected face
for (x,y,w,h) in faces:
    # add bounding box to color image
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

# convert BGR image to RGB for plotting
cv_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# display the image, along with bounding box
plt.imshow(cv_rgb)
plt.show()
```

```
Number of faces detected: 1
```



Before using any of the face detectors, it is standard procedure to convert the images to grayscale. The `detectMultiScale` function executes the classifier stored in `face_cascade` and takes the grayscale image as a parameter.

In the above code, `faces` is a numpy array of detected faces, where each row corresponds to a detected face. Each detected face is a 1D array with four entries that specifies the bounding box of the detected face. The first two entries in the array (extracted in the above code as `x` and `y`) specify the horizontal and vertical positions of the top left corner of the bounding box. The last two entries in the array (extracted here as `w` and `h`) specify the width and height of the box.

1.1.1 Write a Human Face Detector

We can use this procedure to write a function that returns `True` if a human face is detected in an image and `False` otherwise. This function, aptly named `face_detector`, takes a string-valued file path to an image as input and appears in the code block below.

[0]:
```
# returns "True" if face is detected in image stored at img_path
def face_detector(img_path):
    img = cv2.imread(img_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray)
    return len(faces) > 0
```

1.1.2 (IMPLEMENTATION) Assess the Human Face Detector

Question 1: Use the code cell below to test the performance of the `face_detector` function.
- What percentage of the first 100 images in `human_files` have a detected human face?
- What percentage of the first 100 images in `dog_files` have a detected human face?

Ideally, we would like 100% of human images with a detected face and 0% of dog images with a detected face. You will see that our algorithm falls short of this goal, but still gives acceptable performance. We extract the file paths for the first 100 images from each of the datasets and store them in the numpy arrays `human_files_short` and `dog_files_short`.

Answer: (You can print out your results and/or write your percentages in this cell)

99% & 21%

[0]:
```
from tqdm import tqdm

human_files_short = human_files[:100]
dog_files_short = dog_files[:100]

#-#-# Do NOT modify the code above this line. #-#-#

## TODO: Test the performance of the face_detector algorithm
## on the images in human_files_short and dog_files_short.

short = [human_files_short, dog_files_short]

for eachcategory in short:
  score = 0
```

```
  for each in eachcategory:
    if face_detector(each):
      score += 1

  print(score / len(eachcategory))
```

```
0.99
0.21
```

We suggest the face detector from OpenCV as a potential way to detect human images in your algorithm, but you are free to explore other approaches, especially approaches that make use of deep learning :). Please use the code cell below to design and test your own face detection algorithm. If you decide to pursue this *optional* task, report performance on `human_files_short` and `dog_files_short`.

[0]:
```
### (Optional)
### TODO: Test performance of another face detection algorithm.
### Feel free to use as many code cells as needed.
```

Step 2: Detect Dogs

In this section, we use a pre-trained model to detect dogs in images.

1.1.3 Obtain Pre-trained VGG-16 Model

The code cell below downloads the VGG-16 model, along with weights that have been trained on ImageNet, a very large, very popular dataset used for image classification and other vision tasks. ImageNet contains over 10 million URLs, each linking to an image containing an object from one of 1000 categories.

[0]:
```
import torch
import torchvision.models as models

# define VGG16 model
VGG16 = models.vgg16(pretrained=True)

# check if CUDA is available
use_cuda = torch.cuda.is_available()

# move model to GPU if CUDA is available
if use_cuda:
    VGG16 = VGG16.cuda()
```

```
Downloading: "https://download.pytorch.org/models/vgg16-397923af.pth" to
/root/.cache/torch/checkpoints/vgg16-397923af.pth

HBox(children=(IntProgress(value=0, max=553433881), HTML(value='')))
```

Given an image, this pre-trained VGG-16 model returns a prediction (derived from the 1000 possible categories in ImageNet) for the object that is contained in the image.

1.1.4 (IMPLEMENTATION) Making Predictions with a Pre-trained Model

In the next code cell, you will write a function that accepts a path to an image (such as `'dogImages/train/001.Affenpinscher/Affenpinscher_00001.jpg'`) as input and returns the index corresponding to the ImageNet class that is predicted by the pre-trained VGG-16 model. The output should always be an integer between 0 and 999, inclusive.

Before writing the function, make sure that you take the time to learn how to appropriately pre-process tensors for pre-trained models in the PyTorch documentation.

[0]:
```
batch_size = 32
num_workers = 0
```

[0]:
```
from PIL import Image
import torchvision.transforms as transforms

# Set PIL to be tolerant of image files that are truncated.
from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True


normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                 std=[0.229, 0.224, 0.225])

train_transform = transforms.Compose([

                                 transforms.RandomResizedCrop(224),
                                 transforms.RandomVerticalFlip(),
                                 transforms.RandomRotation(10),
                                 transforms.ToTensor(),
                                 normalize])

test_transform = transforms.Compose([
                                 transforms.Resize(255),
                                 transforms.CenterCrop(224),
                                 transforms.ToTensor(),
                                 normalize])

def VGG16_predict(img_path):
    '''
    Use pre-trained VGG-16 model to obtain index corresponding to
    predicted ImageNet class for image at specified path

    Args:
        img_path: path to an image
```

```
    Returns:
        Index corresponding to VGG-16 model's prediction
    '''

    ## TODO: Complete the function.
    ## Load and pre-process an image from the given img_path
    ## Return the *index* of the predicted class for that image
    image = test_transform(Image.open(img_path)).cuda()
    prediction = VGG16(torch.unsqueeze(image, 0)).max(1)[1].item()


    return prediction # predicted class index
```

[0]:

[0]:

[0]:

[0]:

1.1.5 (IMPLEMENTATION) Write a Dog Detector

While looking at the dictionary, you will notice that the categories corresponding to dogs appear in an uninterrupted sequence and correspond to dictionary keys 151-268, inclusive, to include all categories from `'Chihuahua'` to `'Mexican hairless'`. Thus, in order to check to see if an image is predicted to contain a dog by the pre-trained VGG-16 model, we need only check if the pre-trained model predicts an index between 151 and 268 (inclusive).

Use these ideas to complete the `dog_detector` function below, which returns `True` if a dog is detected in an image (and `False` if not).

[0]:
```
### returns "True" if a dog is detected in the image stored at img_path
def dog_detector(img_path):
    pred = VGG16_predict(img_path)

    return(True if pred in range(151, 267) else False) # true/false
```

1.1.6 (IMPLEMENTATION) Assess the Dog Detector

Question 2: Use the code cell below to test the performance of your `dog_detector` function.
- What percentage of the images in `human_files_short` have a detected dog?
- What percentage of the images in `dog_files_short` have a detected dog?

Answer: None of the images in the human files have a detected dog
90% of the images in the dog files has a detected dog

[0]:
```
### TODO: Test the performance of the dog_detector function
### on the images in human_files_short and dog_files_short.
```

```
files = [("Human", human_files_short), ("Dog", dog_files_short)]

for eachcategory in files:
  score = 0

  for each in eachcategory[1]:
    if dog_detector(each):
      score += 1

  print(eachcategory[0], " -- ", score * 100.0 / len(eachcategory[1]))
```

```
Human  --  0.0
Dog  --  90.0
```

We suggest VGG-16 as a potential network to detect dog images in your algorithm, but you are free to explore other pre-trained networks (such as Inception-v3, ResNet-50, etc). Please use the code cell below to test other pre-trained PyTorch models. If you decide to pursue this *optional* task, report performance on `human_files_short` and `dog_files_short`.

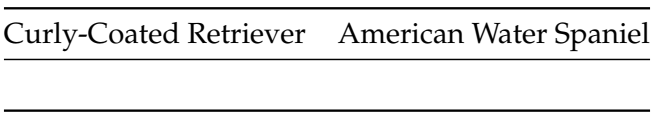
[0]:
```
### (Optional)
### TODO: Report the performance of another pre-trained network.
### Feel free to use as many code cells as needed.
```

Step 3: Create a CNN to Classify Dog Breeds (from Scratch)

Now that we have functions for detecting humans and dogs in images, we need a way to predict breed from images. In this step, you will create a CNN that classifies dog breeds. You must create your CNN *from scratch* (so, you can't use transfer learning *yet*!), and you must attain a test accuracy of at least 10%. In Step 4 of this notebook, you will have the opportunity to use transfer learning to create a CNN that attains greatly improved accuracy.

We mention that the task of assigning breed to dogs from images is considered exceptionally challenging. To see why, consider that *even a human* would have trouble distinguishing between a Brittany and a Welsh Springer Spaniel.

It is not difficult to find other dog breed pairs with minimal inter-class variation (for instance, Curly-Coated Retrievers and American Water Spaniels).



Likewise, recall that labradors come in yellow, chocolate, and black. Your vision-based algorithm will have to conquer this high intra-class variation to determine how to classify all of these

different shades as the same breed.

Yellow Labrador	Chocolate Labrador

We also mention that random chance presents an exceptionally low bar: setting aside the fact that the classes are slightly imabalanced, a random guess will provide a correct answer roughly 1 in 133 times, which corresponds to an accuracy of less than 1%.

Remember that the practice is far ahead of the theory in deep learning. Experiment with many different architectures, and trust your intuition. And, of course, have fun!

1.1.7 (IMPLEMENTATION) Specify Data Loaders for the Dog Dataset

Use the code cell below to write three separate data loaders for the training, validation, and test datasets of dog images (located at `dogImages/train`, `dogImages/valid`, and `dogImages/test`, respectively). You may find this documentation on custom datasets to be a useful resource. If you are interested in augmenting your training and/or validation data, check out the wide variety of transforms!

[0]:
```
import os
from torchvision import datasets

### TODO: Write data loaders for training, validation, and test sets
## Specify appropriate transforms, and batch_sizes

dog_train_dir = '/dog_folder/dogImages/train'
dog_test_dir = '/dog_folder/dogImages/test'
dog_val_dir = '/dog_folder/dogImages/valid'

train_data = datasets.ImageFolder(dog_train_dir, transform=train_transform)
test_data = datasets.ImageFolder(dog_test_dir, transform=test_transform)
val_data = datasets.ImageFolder(dog_val_dir, transform=test_transform)

train_loader = torch.utils.data.DataLoader(train_data, shuffle=True,
                                           batch_size=batch_size,␣
 ↪num_workers=num_workers)
test_loader = torch.utils.data.DataLoader(test_data, shuffle=True,
                                          batch_size=batch_size,␣
 ↪num_workers=num_workers)
val_loader = torch.utils.data.DataLoader(val_data, shuffle=True,
                                         batch_size=batch_size,␣
 ↪num_workers=num_workers)
```

[0]:
```
num_class = 133
```

Question 3: Describe your chosen procedure for preprocessing the data. - How does your code resize the images (by cropping, stretching, etc)? What size did you pick for the input tensor, and why? - Did you decide to augment the dataset? If so, how (through translations, flips, rotations, etc)? If not, why not?

Answer:

The images are resized to 224x224 pixels. This is done in the transformation pipeline. A random resize crop is performed on the training data, and a center crop is done on the test images.

A random rotation as well as a vertical flip is performed on the images. In addition to the random size crop mentioned earlier.

1.1.8 (IMPLEMENTATION) Model Architecture

Create a CNN to classify dog breed. Use the template in the code cell below.

[0]:
```
import torch.nn as nn
import torch.nn.functional as F

# define the CNN architecture
class Net(nn.Module):
    ### TODO: choose an architecture, and complete the class
    def __init__(self):
        super(Net, self).__init__()
        ## Define layers of a CNN
        self.conv1 = nn.Conv2d(3, 16, 3, padding=1)
        self.conv2 = nn.Conv2d(16, 32, 3, padding=1)
        self.conv3 = nn.Conv2d(32, 64, 3, padding=1)
        self.conv4 = nn.Conv2d(64, 128, 3, padding=1)
        self.conv5 = nn.Conv2d(128, 256, 3, padding=1)
        self.conv6 = nn.Conv2d(256, 512, 3, padding=1)
        self.conv7 = nn.Conv2d(512, 1024, 3, padding=1)

        self.pool = nn.MaxPool2d(2, 2)

        self.dropout = nn.Dropout(0.3)

        self.batchnorm16 = nn.BatchNorm2d(16)
        self.batchnorm32 = nn.BatchNorm2d(32)
        self.batchnorm64 = nn.BatchNorm2d(64)
        self.batchnorm128 = nn.BatchNorm2d(128)
        self.batchnorm256 = nn.BatchNorm2d(256)

        self.fc1 = nn.Linear(7*7*1024, 512)
        self.fc2 = nn.Linear(512, num_class)

    def forward(self, x):
        ## Define forward behavior

        # 224x224x3 - 224x224x16
        x = torch.relu(self.conv1(x))
        # 224x224x16 - 112x112x16
        x = self.pool(x)
        x = F.dropout(x)
```

```
        x = self.batchnorm16(x)


        # 112x112x16 - 112x112x32
        x = torch.relu(self.conv2(x))
        # 112x112x32 - 56x56x32
        x = self.pool(x)
        #x = F.dropout(x)
        x = self.batchnorm32(x)


        # 56x56x32 - 56x56x64
        x = torch.relu(self.conv3(x))
        # 56x56x64 - 28x28x64
        x = self.pool(x)
        x = F.dropout(x)
        #x = self.batchnorm64(x)

        # 28x28x64 - 28x28x128
        x = torch.relu(self.conv4(x))
        # 28x28x128 - 14x14x128
        x = self.pool(x)
        #x = F.dropout(x)
        x = self.batchnorm128(x)


        # 14x14x128 - 14x14x256
        x = torch.relu(self.conv5(x))
        # 14x14x256 - 7x7x256
        x = self.pool(x)
        x = F.dropout(x)
        #x = self.batchnorm256(x)

        x = torch.relu(self.conv6(x))
        x = torch.relu(self.conv7(x))


        ## Flatten
        x = x.view(-1, 7*7*1024)

        #x = torch.log_softmax(self.fc1(x), dim=1)
        x = (self.fc1(x))

        return(x)

#-#-# You do NOT have to modify the code below this line. #-#-#
```

```
# instantiate the CNN
model_scratch = Net()

# move tensors to GPU if CUDA is available
if use_cuda:
    model_scratch.cuda()
```

Question 4: Outline the steps you took to get to your final CNN architecture and your reasoning at each step.

Answer:

I found it quite difficult to get to above 10% and had to experiment quite a bit as the training was quite slow. Creating a deeper network of 7 Convolutional Layers with Max-Pooling after the first 5 got me to the final score. I also included Dropout and Batch Normalization after the convolutional layers to reduce overfitting.

1.1.9 (IMPLEMENTATION) Specify Loss Function and Optimizer

Use the next code cell to specify a loss function and optimizer. Save the chosen loss function as `criterion_scratch`, and the optimizer as `optimizer_scratch` below.

[0]:
```
import torch.optim as optim

### TODO: select loss function
criterion_scratch = nn.CrossEntropyLoss()

### TODO: select optimizer
optimizer_scratch = optim.Adagrad(model_scratch.parameters(), lr=0.01)




'''
### TODO: select loss function
criterion_scratch = nn.NLLLoss()

### TODO: select optimizer
optimizer_scratch = optim.Adam(model_scratch.parameters(), lr=0.01)
'''
```

[0]:
```
'\n### TODO: select loss function\ncriterion_scratch = nn.NLLLoss()\n\n### TODO:
select optimizer\noptimizer_scratch = optim.Adam(model_scratch.parameters(),
lr=0.01)\n'
```

[0]:
```
loaders_scratch = {'train' : train_loader,
                   'test': test_loader,
                   'valid': val_loader
                   }
```

1.1.10 (IMPLEMENTATION) Train and Validate the Model

Train and validate your model in the code cell below. Save the final model parameters at filepath `'model_scratch.pt'`.

[0]:
```
# the following import is required for training to be robust to truncated␣
 ↪images
from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True

def train(n_epochs, loaders, model, optimizer, criterion, use_cuda, save_path):
    """returns trained model"""
    # initialize tracker for minimum validation loss
    valid_loss_min = np.Inf

    for epoch in range(1, n_epochs+1):
        # initialize variables to monitor training and validation loss
        train_loss = 0.0
        valid_loss = 0.0

        ###################
        # train the model #
        ###################
        model.train()
        for batch_idx, (data, target) in enumerate(loaders['train']):
            # move to GPU
            if use_cuda:
                data, target = data.cuda(), target.cuda()
            ## find the loss and update the model parameters accordingly
            ## record the average training loss, using something like
            ## train_loss = train_loss + ((1 / (batch_idx + 1)) * (loss.data -␣
 ↪train_loss))

            optimizer.zero_grad()
            outputs = model(data)

            loss = criterion(outputs, target)

            loss.backward()
            optimizer.step()

            train_loss += ((1 / (batch_idx + 1)) * (loss.data - train_loss))
            #print(batch_idx)


        ######################
        # validate the model #
        ######################
```

```
        model.eval()
        for batch_idx, (data, target) in enumerate(loaders['valid']):
            # move to GPU
            if use_cuda:
                data, target = data.cuda(), target.cuda()
            ## update the average validation loss
            outputs = model(data)

            loss = criterion(outputs, target)
            valid_loss += ((1 / (batch_idx + 1)) * (loss.data - valid_loss))

        # print training/validation statistics
        print('Epoch: {} \tTraining Loss: {:.6f} \tValidation Loss: {:.6f}'.
↪format(
            epoch,
            train_loss,
            valid_loss
            ))

        ## TODO: save the model if validation loss has decreased
        if valid_loss <= valid_loss_min:
          torch.save(model.state_dict(), save_path)
          valid_loss_min = valid_loss
          print("Validation Loss Reduced ", valid_loss.data, " - ",
                valid_loss_min, " Model saved")
    # return trained model
    return model
```

[0]:
```
# train the model
model_scratch = train(100, loaders_scratch, model_scratch, optimizer_scratch,
                      criterion_scratch, use_cuda, 'model_scratch.pt')
```

```
Epoch: 1        Training Loss: 4.086291         Validation Loss: 4.186299
Validation Loss Reduced  tensor(4.1863, device='cuda:0')  -  tensor(4.1863,
device='cuda:0')  Model saved
Epoch: 2        Training Loss: 4.068234         Validation Loss: 4.211978
Epoch: 3        Training Loss: 4.045761         Validation Loss: 4.223539
Epoch: 4        Training Loss: 4.053134         Validation Loss: 4.188251
Epoch: 5        Training Loss: 4.048547         Validation Loss: 4.133544
Validation Loss Reduced  tensor(4.1335, device='cuda:0')  -  tensor(4.1335,
device='cuda:0')  Model saved
Epoch: 6        Training Loss: 4.040422         Validation Loss: 4.143997
Epoch: 7        Training Loss: 4.010637         Validation Loss: 4.182335
Epoch: 8        Training Loss: 4.023798         Validation Loss: 4.181041
Epoch: 9        Training Loss: 3.997694         Validation Loss: 4.176572
Epoch: 10       Training Loss: 4.006224         Validation Loss: 4.168719
Epoch: 11       Training Loss: 4.002428         Validation Loss: 4.155929
```

```
Epoch: 12       Training Loss: 3.985068         Validation Loss: 4.154250
Epoch: 13       Training Loss: 3.963163         Validation Loss: 4.161075
Epoch: 14       Training Loss: 3.964175         Validation Loss: 4.180617
Epoch: 15       Training Loss: 3.966568         Validation Loss: 4.157337
Epoch: 16       Training Loss: 3.944619         Validation Loss: 4.198622
Epoch: 17       Training Loss: 3.947421         Validation Loss: 4.212296
Epoch: 18       Training Loss: 3.916632         Validation Loss: 4.148021
Epoch: 19       Training Loss: 3.937975         Validation Loss: 4.111863
Validation Loss Reduced  tensor(4.1119, device='cuda:0')  -  tensor(4.1119,
device='cuda:0')  Model saved
Epoch: 20       Training Loss: 3.925767         Validation Loss: 4.182833
Epoch: 21       Training Loss: 3.915854         Validation Loss: 4.118954
Epoch: 22       Training Loss: 3.923850         Validation Loss: 4.151489
Epoch: 23       Training Loss: 3.891236         Validation Loss: 4.139501
Epoch: 24       Training Loss: 3.888993         Validation Loss: 4.121121
Epoch: 25       Training Loss: 3.874236         Validation Loss: 4.117653
Epoch: 26       Training Loss: 3.879977         Validation Loss: 4.102843
Validation Loss Reduced  tensor(4.1028, device='cuda:0')  -  tensor(4.1028,
device='cuda:0')  Model saved
Epoch: 27       Training Loss: 3.857108         Validation Loss: 4.094849
Validation Loss Reduced  tensor(4.0948, device='cuda:0')  -  tensor(4.0948,
device='cuda:0')  Model saved
Epoch: 28       Training Loss: 3.866122         Validation Loss: 4.161791
Epoch: 29       Training Loss: 3.815815         Validation Loss: 4.092135
Validation Loss Reduced  tensor(4.0921, device='cuda:0')  -  tensor(4.0921,
device='cuda:0')  Model saved
Epoch: 30       Training Loss: 3.833266         Validation Loss: 4.160854
Epoch: 31       Training Loss: 3.849227         Validation Loss: 4.081587
Validation Loss Reduced  tensor(4.0816, device='cuda:0')  -  tensor(4.0816,
device='cuda:0')  Model saved
Epoch: 32       Training Loss: 3.835896         Validation Loss: 4.158099
Epoch: 33       Training Loss: 3.803946         Validation Loss: 4.116474
Epoch: 34       Training Loss: 3.807491         Validation Loss: 4.114796
Epoch: 35       Training Loss: 3.787673         Validation Loss: 4.127579
Epoch: 36       Training Loss: 3.799158         Validation Loss: 4.095510
Epoch: 37       Training Loss: 3.786133         Validation Loss: 4.040776
Validation Loss Reduced  tensor(4.0408, device='cuda:0')  -  tensor(4.0408,
device='cuda:0')  Model saved
Epoch: 38       Training Loss: 3.770112         Validation Loss: 4.123898
Epoch: 39       Training Loss: 3.797170         Validation Loss: 4.096382
Epoch: 40       Training Loss: 3.746605         Validation Loss: 4.096305
Epoch: 41       Training Loss: 3.769789         Validation Loss: 4.123235
Epoch: 42       Training Loss: 3.732310         Validation Loss: 4.178492
Epoch: 43       Training Loss: 3.718544         Validation Loss: 4.059054
Epoch: 44       Training Loss: 3.711756         Validation Loss: 4.030836
Validation Loss Reduced  tensor(4.0308, device='cuda:0')  -  tensor(4.0308,
device='cuda:0')  Model saved
Epoch: 45       Training Loss: 3.701263         Validation Loss: 4.153516
```

```
Epoch: 46       Training Loss: 3.680897         Validation Loss: 4.112167
Epoch: 47       Training Loss: 3.710083         Validation Loss: 4.106149
Epoch: 48       Training Loss: 3.705883         Validation Loss: 4.114625
Epoch: 49       Training Loss: 3.678161         Validation Loss: 4.159655
Epoch: 50       Training Loss: 3.671437         Validation Loss: 4.149263
Epoch: 51       Training Loss: 3.664629         Validation Loss: 4.114591
Epoch: 52       Training Loss: 3.647309         Validation Loss: 4.162127
Epoch: 53       Training Loss: 3.644874         Validation Loss: 4.149492
Epoch: 54       Training Loss: 3.628225         Validation Loss: 4.030641
Validation Loss Reduced  tensor(4.0306, device='cuda:0')  -  tensor(4.0306,
device='cuda:0')  Model saved
Epoch: 55       Training Loss: 3.645099         Validation Loss: 4.161564
Epoch: 56       Training Loss: 3.652722         Validation Loss: 4.104355
Epoch: 57       Training Loss: 3.594668         Validation Loss: 4.132323
Epoch: 58       Training Loss: 3.626989         Validation Loss: 4.128347
Epoch: 59       Training Loss: 3.602898         Validation Loss: 4.162079
Epoch: 60       Training Loss: 3.606078         Validation Loss: 4.108014
Epoch: 61       Training Loss: 3.604295         Validation Loss: 4.039760
Epoch: 62       Training Loss: 3.571082         Validation Loss: 4.104225
Epoch: 63       Training Loss: 3.570262         Validation Loss: 4.103229
Epoch: 64       Training Loss: 3.562581         Validation Loss: 4.160910
Epoch: 65       Training Loss: 3.547214         Validation Loss: 4.165713
Epoch: 66       Training Loss: 3.561982         Validation Loss: 4.070630
Epoch: 67       Training Loss: 3.557245         Validation Loss: 4.196130
Epoch: 68       Training Loss: 3.579003         Validation Loss: 4.082671
Epoch: 69       Training Loss: 3.544309         Validation Loss: 4.155773
Epoch: 70       Training Loss: 3.513943         Validation Loss: 4.067127
Epoch: 71       Training Loss: 3.509916         Validation Loss: 4.135667
Epoch: 72       Training Loss: 3.525071         Validation Loss: 4.129495
Epoch: 73       Training Loss: 3.506387         Validation Loss: 4.124705
Epoch: 74       Training Loss: 3.476527         Validation Loss: 4.099450
Epoch: 75       Training Loss: 3.474761         Validation Loss: 4.179624
Epoch: 76       Training Loss: 3.474026         Validation Loss: 4.100056
Epoch: 77       Training Loss: 3.471897         Validation Loss: 4.054889
Epoch: 78       Training Loss: 3.436900         Validation Loss: 4.106069
Epoch: 79       Training Loss: 3.474981         Validation Loss: 4.109394
Epoch: 80       Training Loss: 3.455547         Validation Loss: 4.091047
Epoch: 81       Training Loss: 3.436061         Validation Loss: 4.113472
Epoch: 82       Training Loss: 3.415712         Validation Loss: 4.166812
Epoch: 83       Training Loss: 3.434718         Validation Loss: 4.103451
Epoch: 84       Training Loss: 3.385183         Validation Loss: 4.109819
Epoch: 85       Training Loss: 3.400118         Validation Loss: 4.135925
Epoch: 86       Training Loss: 3.448308         Validation Loss: 4.254159
Epoch: 87       Training Loss: 3.387196         Validation Loss: 4.228027
Epoch: 88       Training Loss: 3.389344         Validation Loss: 4.171871
Epoch: 89       Training Loss: 3.386137         Validation Loss: 4.132946
Epoch: 90       Training Loss: 3.382780         Validation Loss: 4.250198
Epoch: 91       Training Loss: 3.347989         Validation Loss: 4.234111
```

```
Epoch: 92       Training Loss: 3.346647         Validation Loss: 4.135743
Epoch: 93       Training Loss: 3.356438         Validation Loss: 4.132249
Epoch: 94       Training Loss: 3.357662         Validation Loss: 4.101941
Epoch: 95       Training Loss: 3.330262         Validation Loss: 4.107337
Epoch: 96       Training Loss: 3.324895         Validation Loss: 4.168148
Epoch: 97       Training Loss: 3.303341         Validation Loss: 4.215924
Epoch: 98       Training Loss: 3.324808         Validation Loss: 4.179361
Epoch: 99       Training Loss: 3.310668         Validation Loss: 4.073009
Epoch: 100      Training Loss: 3.317271         Validation Loss: 4.218546
```

[0]:
```
# load the model that got the best validation accuracy
model_scratch.load_state_dict(torch.load('model_scratch.pt'))
```

[0]:
```
model_scratch
```

[0]:
```
Net(
  (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv4): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv5): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv6): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv7): Conv2d(512, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  (dropout): Dropout(p=0.3, inplace=False)
  (batchnorm16): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (batchnorm32): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (batchnorm64): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (batchnorm128): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (batchnorm256): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (fc1): Linear(in_features=50176, out_features=512, bias=True)
  (fc2): Linear(in_features=512, out_features=133, bias=True)
)
```

[0]:
```
torch.save(model_scratch.state_dict(), 'model_scratch.pt')
```

1.1.11 (IMPLEMENTATION) Test the Model

Try out your model on the test dataset of dog images. Use the code cell below to calculate and print the test loss and accuracy. Ensure that your test accuracy is greater than 10%.

[0]:
```
def test(loaders, model, criterion, use_cuda):

```

```
    # monitor test loss and accuracy
    test_loss = 0.
    correct = 0.
    total = 0.

    model.eval()
    for batch_idx, (data, target) in enumerate(loaders['test']):
        # move to GPU
        if use_cuda:
            data, target = data.cuda(), target.cuda()
        # forward pass: compute predicted outputs by passing inputs to the␣
↪model
        output = model(data)
        # calculate the loss
        loss = criterion(output, target)
        # update average test loss
        test_loss = test_loss + ((1 / (batch_idx + 1)) * (loss.data -␣
↪test_loss))
        # convert output probabilities to predicted class
        pred = output.data.max(1, keepdim=True)[1]
        # compare predictions to true label
        correct += np.sum(np.squeeze(pred.eq(target.data.view_as(pred))).cpu().
↪numpy())
        total += data.size(0)

    print('Test Loss: {:.6f}\n'.format(test_loss))

    print('\nTest Accuracy: %2d%% (%2d/%2d)' % (
        100. * correct / total, correct, total))

# call test function
test(loaders_scratch, model_scratch, criterion_scratch, use_cuda)
```

```
Test Loss: 3.961052


Test Accuracy: 10% (89/836)
```

Step 4: Create a CNN to Classify Dog Breeds (using Transfer Learning)

You will now use transfer learning to create a CNN that can identify dog breed from images. Your CNN must attain at least 60% accuracy on the test set.

1.1.12 (IMPLEMENTATION) Specify Data Loaders for the Dog Dataset

Use the code cell below to write three separate data loaders for the training, validation, and test datasets of dog images (located at `dogImages/train`, `dogImages/valid`, and `dogImages/test`, respectively).

If you like, **you are welcome to use the same data loaders from the previous step**, when you created a CNN from scratch.

[0]:
```
## TODO: Specify data loaders
```

1.1.13 (IMPLEMENTATION) Model Architecture

Use transfer learning to create a CNN to classify dog breed. Use the code cell below, and save your initialized model as the variable `model_transfer`.

[0]:
```
import torchvision.models as models
import torch.nn as nn

## TODO: Specify model architecture

VGG19 = models.vgg19(pretrained=True)


# Freeze training for all "features" layers
for param in VGG19.features.parameters():
    param.requires_grad = False


VGG19.classifier = nn.Sequential(nn.Linear(25088, num_class))


use_cuda = torch.cuda.is_available()



# move tensors to GPU if CUDA is available
if use_cuda:
    VGG19.cuda()
```

[0]:
```
VGG19
```

[0]:
```
VGG(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU(inplace=True)
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU(inplace=True)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
```

```
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU(inplace=True)
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU(inplace=True)
    (16): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (17): ReLU(inplace=True)
    (18): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (19): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20): ReLU(inplace=True)
    (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22): ReLU(inplace=True)
    (23): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (24): ReLU(inplace=True)
    (25): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (26): ReLU(inplace=True)
    (27): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (29): ReLU(inplace=True)
    (30): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (31): ReLU(inplace=True)
    (32): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (33): ReLU(inplace=True)
    (34): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (35): ReLU(inplace=True)
    (36): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
  (classifier): Sequential(
    (0): Linear(in_features=25088, out_features=133, bias=True)
  )
)
```

Question 5: Outline the steps you took to get to your final CNN architecture and your reasoning at each step. Describe why you think the architecture is suitable for the current problem.

Answer: The VGG19 model was selected because it's a deeper network than the VGG16 which we already know performs quite well at predicting different dog breeds. The classifier part only consists of one linear layer for simplicity and speedy training.

[0]:

1.1.14 (IMPLEMENTATION) Specify Loss Function and Optimizer

Use the next code cell to specify a loss function and optimizer. Save the chosen loss function as `criterion_transfer`, and the optimizer as `optimizer_transfer` below.

[0]:
```
criterion_transfer = nn.CrossEntropyLoss()
optimizer_transfer = optim.Adagrad(VGG19.parameters(), lr=0.01)
```

[0]:
```
loaders_transfer = {'train' : train_loader,
                    'test': test_loader,
                    'valid': val_loader
                    }
```

1.1.15 (IMPLEMENTATION) Train and Validate the Model

Train and validate your model in the code cell below. Save the final model parameters at filepath `'model_transfer.pt'`.

[0]:
```
# train the model
#model_transfer = # train(n_epochs, loaders_transfer, model_transfer,␣
 ↪optimizer_transfer, criterion_transfer, use_cuda, 'model_transfer.pt')
model_transfer = train(10, loaders_transfer, VGG19, optimizer_transfer,
                       criterion_transfer, use_cuda, 'model_transfer_scratch.
 ↪pt')
```

```
Epoch: 1        Training Loss: 14.871022        Validation Loss: 5.201969
Validation Loss Reduced  tensor(5.2020, device='cuda:0')  -  tensor(5.2020,
device='cuda:0')  Model saved
Epoch: 2        Training Loss: 5.798612         Validation Loss: 4.022574
Validation Loss Reduced  tensor(4.0226, device='cuda:0')  -  tensor(4.0226,
device='cuda:0')  Model saved
Epoch: 3        Training Loss: 4.468754         Validation Loss: 3.193850
Validation Loss Reduced  tensor(3.1938, device='cuda:0')  -  tensor(3.1938,
device='cuda:0')  Model saved
Epoch: 4        Training Loss: 3.873490         Validation Loss: 3.309093
Epoch: 5        Training Loss: 3.448864         Validation Loss: 2.636044
Validation Loss Reduced  tensor(2.6360, device='cuda:0')  -  tensor(2.6360,
device='cuda:0')  Model saved
Epoch: 6        Training Loss: 3.194677         Validation Loss: 2.562018
Validation Loss Reduced  tensor(2.5620, device='cuda:0')  -  tensor(2.5620,
device='cuda:0')  Model saved
Epoch: 7        Training Loss: 2.868114         Validation Loss: 2.285780
Validation Loss Reduced  tensor(2.2858, device='cuda:0')  -  tensor(2.2858,
device='cuda:0')  Model saved
Epoch: 8        Training Loss: 2.709471         Validation Loss: 2.355555
Epoch: 9        Training Loss: 2.524587         Validation Loss: 2.384459
Epoch: 10       Training Loss: 2.488919         Validation Loss: 1.991625
Validation Loss Reduced  tensor(1.9916, device='cuda:0')  -  tensor(1.9916,
device='cuda:0')  Model saved
```

[0]:
```
# load the model that got the best validation accuracy (uncomment the line␣
↪below)
model_transfer.load_state_dict(torch.load('model_transfer_scratch.pt'))
```

[0]:
```
<All keys matched successfully>
```

1.1.16 (IMPLEMENTATION) Test the Model

Try out your model on the test dataset of dog images. Use the code cell below to calculate and print the test loss and accuracy. Ensure that your test accuracy is greater than 60%.

[0]:
```
test(loaders_transfer, model_transfer, criterion_transfer, use_cuda)
```

```
Test Loss: 2.192559


Test Accuracy: 76% (636/836)
```

1.1.17 (IMPLEMENTATION) Predict Dog Breed with the Model

Write a function that takes an image path as input and returns the dog breed (`Affenpinscher`, `Afghan hound`, etc) that is predicted by your model.

[0]:
```
### TODO: Write a function that takes a path to an image as input
### and returns the dog breed that is predicted by the model.

data_transfer = {'train' : train_data, 'valid': val_data, 'test': test_data}


# list of class names by index, i.e. a name can be accessed like class_names[0]
class_names = [item[4:].replace("_", " ") for item in data_transfer['train'].
↪classes]

def predict_breed_transfer(img_path):
    # load the image and return the predicted breed

    image = test_transform(Image.open(img_path)).cuda()
    prediction = model_transfer(torch.unsqueeze(image, 0)).max(1)[1].item()

    return class_names[prediction]
```

Step 5: Write your Algorithm

Write an algorithm that accepts a file path to an image and first determines whether the image contains a human, dog, or neither. Then, - if a **dog** is detected in the image, return the predicted breed. - if a **human** is detected in the image, return the resembling dog breed. - if **neither** is detected in the image, provide output that indicates an error.