PokeNet

June 25, 2020

[1]:
```
# Importing all necessary libraries
import numpy as np
import pandas as pd
import cv2 as cv # requires manual install to Anaconda
import os
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
import warnings
import gc
import requests
import random
from icrawler.builtin import GoogleImageCrawler # requires manual install to␣
 ↪Anaconda
from sklearn.model_selection import train_test_split

# Keras, Tensorflow stuff for CNN/VGG
from keras.utils import to_categorical
from keras.preprocessing.image import ImageDataGenerator
from keras import backend as K
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import GlobalAveragePooling2D, Lambda, Conv2D, MaxPooling2D,␣
 ↪Dropout, Dense, Flatten, Activation
from keras.layers.normalization import BatchNormalization
from keras.optimizers import SGD, RMSprop, Adam

# plot style preferences
style.use('fivethirtyeight')
sns.set_style("darkgrid")

%matplotlib inline
```

```
Using TensorFlow backend.
```

[197]:
```
# Crawl for images of specified pokemon
pokemon = ['Pikachu', 'Squirtle', 'Bulbasaur', 'Charmander', 'Mewtwo']
```

```
for i in pokemon:
    crawler = GoogleImageCrawler(storage={'root_dir': 'dataset/' + i})
    crawler.crawl(keyword=i, max_num=500, min_size=(96,96))

# ran this later to get detective pikachu images from google
# have to manual go through images to get outliers and then add the good ones
# to our main dataset manually
for i in pokemon:
    crawler = GoogleImageCrawler(storage={'root_dir': 'outliers/' + i})
    crawler.crawl(keyword='Detective Pikachu ' + i, max_num=100,␣
 ↪min_size=(96,96))
```

```
2020-06-25 00:54:16,335 - INFO - icrawler.crawler - start crawling…
2020-06-25 00:54:16,336 - INFO - icrawler.crawler - starting 1 feeder threads…
2020-06-25 00:54:16,337 - INFO - feeder - thread feeder-001 exit
2020-06-25 00:54:16,337 - INFO - icrawler.crawler - starting 1 parser threads…
2020-06-25 00:54:16,339 - INFO - icrawler.crawler - starting 1 downloader
threads…
2020-06-25 00:54:16,919 - INFO - parser - parsing result page https://www.google
.com/search?q=Detective+Pikachu+Squirtle&ijn=0&start=0&tbs=&tbm=isch
2020-06-25 00:54:17,095 - INFO - downloader - image #1
https://i.ytimg.com/vi/wbFt4PpwggQ/maxresdefault.jpg
2020-06-25 00:54:17,716 - INFO - downloader - image #2  https://cdn.vox-cdn.com/
thumbor/7CFvl9b0UpEQviCDY0RfXzHBoZ0=/1400x0/filters:no_upscale()/cdn.vox-cdn.com
/uploads/chorus_asset/file/16023556/Screen_Shot_2019_04_10_at_5.06.16_PM.png
2020-06-25 00:54:17,752 - INFO - downloader - image #3
https://i.ytimg.com/vi/DPE3J9ama_E/maxresdefault.jpg
2020-06-25 00:54:17,913 - INFO - downloader - image #4  https://images-wixmp-ed3
0a86b8c4ca887773594c2.wixmp.com/f/295fb76c-7179-4c70-a508-a1cce61a876f/dd4aps3-5
c7a8849-aab1-4f8f-933e-84240f3c1c35.png?token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1Ni
J9.eyJzdWIiOiJ1cm46YXBwOiIsImlzcyI6InVybjphcHA6Iiwib2JqIjpbW3sicGF0aCI6IlwvZlwvM
jk1ZmI3NmMtNzE3OS00YzcwLWE1MDgtYTFjY2U2MWE4NzZmXC9kZDRhcHMzLTVjN2E4ODQ5LWFhYjEtN
GY4ZiO5MzNlLTg0MjQwZjNjMWMzNS5wbmcifV1dLCJhdWQiOlsidXJuOnNlcnZpY2U6ZmlsZS5kb3dub
G9hZCJdfQ.yolWSOEufTtHvpjGsDVvkogEkh8gJSk7EaYolU4_rsM
2020-06-25 00:54:18,029 - INFO - downloader - image #5
https://cdna.artstation.com/p/assets/images/images/017/754/158/large/julie-
tardieu-06.jpg?1557228628
2020-06-25 00:54:18,265 - INFO - downloader - image #6
https://www.thehdroom.com/wp-content/uploads/2019/04/new-detective-pikachu-
footage.jpg
2020-06-25 00:54:18,621 - INFO - downloader - image #7
https://cdn1.thr.com/sites/default/files/2019/05/detective_pikachu-bulbasaur-
publicity-h_2019.jpg
2020-06-25 00:54:18,641 - INFO - downloader - image #8  https://cdn.vox-cdn.com/
thumbor/g0daWh_TciOX-n5x5viFl0pllmE=/1400x1400/filters:format(png)/cdn.vox-cdn.c
om/uploads/chorus_asset/file/16022665/Screen_Shot_2019_04_10_at_12.26.32_PM.png
2020-06-25 00:54:18,907 - INFO - downloader - image #9  https://nerdist.com/wp-
```

content/uploads/2019/05/Detective-Pikachu-casting-1200x676.jpg
2020-06-25 00:54:19,100 - INFO - downloader - image #10
https://cdn.dribbble.com/users/2367883/screenshots/6477123/shot-
cropped-1557766487823.png
2020-06-25 00:54:19,379 - INFO - downloader - image #11
https://thepopinsider.com/wp-
content/uploads/2019/04/DetectivePikachu_Squirtle.jpg
2020-06-25 00:54:20,120 - INFO - downloader - image #12
http://ricedigital.co.uk/wp-content/uploads/2019/05/Squirtle-555x350.jpg
2020-06-25 00:54:20,929 - INFO - downloader - image #13
https://i.pinimg.com/originals/95/91/34/95913461a301b5215953a491ad57bcb1.gif
2020-06-25 00:54:22,236 - INFO - downloader - image #14 https://66.media.tumblr.
com/6ac780a38a8d9eafed01daf41d6cd45b/tumblr_ps1sykZPEV1vplr8j_400.png
2020-06-25 00:54:22,637 - INFO - downloader - image #15
https://www.thesun.co.uk/wp-content/uploads/2019/04/asc-composite-
detectivepikachu.jpg?strip=all&quality=100&w=1200&h=800&crop=1
2020-06-25 00:54:22,810 - INFO - downloader - image #16 https://external-preview
.redd.it/9XiADhPsTauolR5moAjku0g7oi2Gm1TZ1dB9dlqSjRc.jpg?auto=webp&s=5d6316c1478
5f103d838b5ca1d7e8bb0a868bafa
2020-06-25 00:54:24,014 - INFO - downloader - image #17 https://www.channelnewsa
sia.com/image/11313830/1x1/600/600/303b5411eb34a1eaeed382ebde634b24/Bg/pokemon.j
pg
2020-06-25 00:54:24,174 - INFO - downloader - image #18
https://s3.amazonaws.com/prod-
media.gameinformer.com/styles/full/s3/2018/11/12/77db3cda/pikachumovie.jpg
2020-06-25 00:54:24,414 - INFO - downloader - image #19
https://i.redd.it/ce5pubg191u21.jpg
2020-06-25 00:54:24,497 - INFO - downloader - image #20 https://cdn.vox-cdn.com/
thumbor/sQXDUl49gchks_fgxkqvaXji4pI=/1400x1400/filters:format(jpeg)/cdn.vox-
cdn.com/uploads/chorus_asset/file/13430765/detective_pikachu_trailer_24_1920.jpg
2020-06-25 00:54:24,607 - INFO - downloader - image #21
https://d13ezvd6yrslxm.cloudfront.net/wp/wp-content/images/detective-pikachu-
poster-cropped-700x339.jpeg
2020-06-25 00:54:25,204 - INFO - downloader - image #22 https://poketouch.files.
wordpress.com/2018/11/pokemon_detective_pikachu_movie_bulbasaur_morelull_and_jus
tice_smith_holding_pikachu.png
2020-06-25 00:54:25,973 - INFO - downloader - image #23 https://imagenes.milenio
.com/Byd9it6e0jOav0llTLNEsLNZgb8=/958x596/https://www.milenio.com/uploads/media/
2019/04/10/el-meme-vamo-a-calmano_23_0_1607_1000.jpg
2020-06-25 00:54:26,152 - INFO - downloader - image #24
https://cdn.images.express.co.uk/img/dynamic/36/750x445/1044754.jpg
2020-06-25 00:54:26,288 - INFO - downloader - image #25
https://i.ytimg.com/vi/M3E_g0z9Cys/maxresdefault.jpg
2020-06-25 00:54:26,647 - INFO - downloader - image #26
https://static0.srcdn.com/wordpress/wp-content/uploads/2018/11/Detective-
Pikachu-Squirtle.jpg
2020-06-25 00:54:26,935 - INFO - downloader - image #27
https://i.imgur.com/misRiiZ.png

```
2020-06-25 00:54:27,042 - INFO - downloader - image #28
https://cdn.mos.cms.futurecdn.net/MphYS8aibgiFZVT3de4bSV.jpg
2020-06-25 00:54:27,108 - INFO - downloader - image #29
https://i.redd.it/yh6woxws11y11.png
2020-06-25 00:54:27,527 - INFO - downloader - image #30
https://static1.srcdn.com/wordpress/wp-content/uploads/2018/11/Detective-
Pikachu-Every-Pokemon.jpg
2020-06-25 00:54:28,071 - INFO - downloader - image #31
https://miro.medium.com/max/1200/1*pIhWVOtVAJDUmUJP1jRtVQ.png
2020-06-25 00:54:28,246 - INFO - downloader - image #32
https://media.comicbook.com/2018/11/detective-pikachu-starter-
header-1143895-1280x0.jpeg
2020-06-25 00:54:28,320 - INFO - downloader - image #33 https://cdn.vox-cdn.com/
thumbor/Itoov8zegYkBSrG7ea35aU82bc8=/1400x0/filters:no_upscale()/cdn.vox-cdn.com
/uploads/chorus_asset/file/16023618/Screen_Shot_2019_04_10_at_5.19.56_PM.png
2020-06-25 00:54:28,464 - ERROR - downloader - Response status code 404, file
http://mouse.latercera.com/wp-content/uploads/2019/05/detective-pikachu-1.jpg
2020-06-25 00:54:28,582 - INFO - downloader - image #34
https://d1lss44hh2trtw.cloudfront.net/assets/article/2019/02/26/all-pokemon-in-
detective-pikachu_feature.jpg
2020-06-25 00:54:34,696 - ERROR - downloader - Exception caught when downloading
file https://sm.ign.com/t/ign_ap/gallery/e/every-poke/every-pokemon-in-the-
detective-pikachu-movie_pexr.1080.jpg, error:
HTTPSConnectionPool(host='sm.ign.com', port=443): Read timed out. (read
timeout=5), remaining retry times: 2
2020-06-25 00:54:35,358 - INFO - downloader - image #35
https://sm.ign.com/t/ign_ap/gallery/e/every-poke/every-pokemon-in-the-detective-
pikachu-movie_pexr.1080.jpg
2020-06-25 00:54:35,476 - INFO - downloader - image #36
https://pbs.twimg.com/media/D3zpHLvU0AE3s5b.jpg
2020-06-25 00:54:36,012 - INFO - downloader - image #37 https://nerdist.com/wp-
content/uploads/2019/02/Detective-Pikachu-Braviary.png
2020-06-25 00:54:36,450 - INFO - downloader - image #38
https://movies.mxdwn.com/wp-content/uploads/2019/04/download.jpg
2020-06-25 00:54:36,507 - INFO - downloader - image #39 https://cdn.vox-cdn.com/
thumbor/v2U85x7stV62ecUKZmjqSjpPzNY=/0x0:1024x500/1200x800/filters:focal(236x289
:398x451)/cdn.vox-cdn.com/uploads/chorus_image/image/63721566/Banner.0.jpg
2020-06-25 00:54:36,661 - INFO - downloader - image #40
https://ae01.alicdn.com/kf/Haffd9ac9cbe841f2881061343a89f883x/Takara-Tomy-
Pokemon-Detective-pikachu-Psyduck-Mewtwo-Bulbasaur-Squirtle-Eevee-anime-action-
toy-figures-model.jpg_q50.jpg
2020-06-25 00:54:36,892 - INFO - downloader - image #41
https://media.distractify.com/brand-img/I5aSU-EaQ/480x252/detective-
pikachu-1557498103339.jpg
2020-06-25 00:54:36,948 - INFO - downloader - image #42
https://i.imgur.com/GERfojf.jpg?fb
2020-06-25 00:54:37,048 - INFO - downloader - image #43 https://pyxis.nymag.com/
v1/imgs/4a9/af5/e736d91cd589d1fe2545d1bb1d00b7ba95-08-psyduck.rsquare.w700.jpg
```

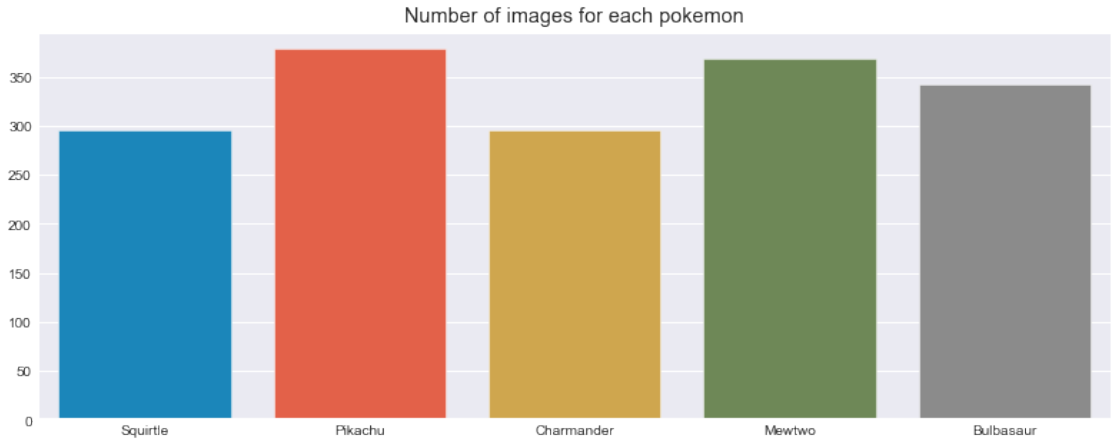
```
2020-06-25 00:54:37,125 - INFO - downloader - image #44
https://d1lss44hh2trtw.cloudfront.net/assets/editorial/2019/02/det-pikachu-
bulbasaur.jpg
2020-06-25 00:54:37,383 - INFO - downloader - image #45 https://images-wixmp-ed3
0a86b8c4ca887773594c2.wixmp.com/f/295fb76c-7179-4c70-a508-a1cce61a876f/dcrxray-e
f618793-8f15-4bee-b1b2-250d0808a43b.png?token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1Ni
J9.eyJzdWIiOiJ1cm46YXBwOiIsImlzcyI6InVybjphcHA6Iiwib2JqIjpbW3sicGF0aCI6IlwvZlwvM
jk1ZmI3NmMtNzE3OS00YzcwLWE1MDgtYTFjY2U2MWE4NzZmXC9kY3J4cmF5LWVmNjE4NzkzLThmMTUtN
GJlZS1iMWIyLTI1MGQwODA4YTQzYi5wbmcifV1dLCJhdWQiOlsidXJuOnNlcnZpY2U6ZmlsZS5kb3dub
G9hZCJdfQ.gIMfBE21wUopFBooMZ-DPOb4hFD8sXmQoisRTGM4nNQ
2020-06-25 00:54:37,789 - INFO - downloader - image #46
https://en.freegames66.com/wp-
content/uploads/posts/77eb10760b8342249f490b3ee8e62a64.jpg
2020-06-25 00:54:37,948 - INFO - downloader - image #47 https://www.channelnewsa
sia.com/image/11312896/1x1/600/600/5f39060743064ad8d1928bab67390c09/ge/snorlax-
pokemon-detective-pikachu.png
2020-06-25 00:54:38,306 - INFO - downloader - image #48
https://static1.srcdn.com/wordpress/wp-content/uploads/2019/02/Detective-
Pikachu-Pokemon-Header.jpg
2020-06-25 00:54:38,416 - INFO - downloader - image #49 https://img.ifunny.co/im
ages/8f875b92c29da85d0db757abb6b9dfd793e72846880d56e3cb62bb61ea510ca0_1.jpg
2020-06-25 00:54:38,440 - INFO - downloader - image #50 https://pyxis.nymag.com/
v1/imgs/635/968/cdd9e17681dc850b2dee6eb0f51d667313-06-detective-
pikachu.rsquare.w700.jpg
2020-06-25 00:54:38,791 - INFO - downloader - downloaded images reach max num,
thread downloader-001 is ready to exit
2020-06-25 00:54:38,792 - INFO - downloader - thread downloader-001 exit
2020-06-25 00:54:39,407 - INFO - icrawler.crawler - Crawling task done!
2020-06-25 00:54:39,444 - INFO - parser - downloaded image reached max num,
thread parser-001 is ready to exit
2020-06-25 00:54:39,445 - INFO - parser - thread parser-001 exit
```

[2]:
```
#constants
DATA_DIR = '/Users/brandonreid/Science/pokemon'
IMG_SIZE = 96
LEARN_RATE = 1e-4
IMAGE_DIMS = (IMG_SIZE, IMG_SIZE, 3)
EPOCHS = 100
SEED = 42
FIG_SIZE = (12, 5)
BS = 32
```

[3]:
```
# obtain number of images for each pokemon
pokemon = os.listdir(DATA_DIR)
print('Total number of pokemon:', len(pokemon))

images = {}
```

```
for poke in pokemon:
    images[poke] = len(os.listdir(os.path.join(DATA_DIR, poke)))

print('Total number of images:', sum(list(images.values())))

classes = list(images.keys())
counts = list(images.values())

fig = plt.figure(figsize = FIG_SIZE)
sns.barplot(x=classes, y=counts).set_title('Number of images for each pokemon')
plt.show()
```

```
Total number of pokemon: 5
Total number of images: 1679
```



[4]:
```
data = [] # List for data (images)
labels = [] # List for data labels

# Loop through pokemon classes
for poke in pokemon:
    path = os.path.join(DATA_DIR, poke) # dir of pokemon

    # randomize image paths before resizing them
    image_paths = sorted(list(os.listdir(path)))
    random.seed(SEED)
    random.shuffle(image_paths)

    # loop over each image of pokemon
    for img in image_paths:
        image = cv.imread(os.path.join(path, img))
        try:
```

```
            data.append(cv.resize(image, (IMG_SIZE, IMG_SIZE))) # add resized␣
 ↪image to dataset
            labels.append(pokemon.index(poke)) # add label as index of pokemon
        except: # error handling for unreadable images
            print(os.path.join(path, img), '-> UNREADABLE')
            continue

print(len(data), " TOTAL IMAGES PROCESSED")
```

```
/Users/brandonreid/Science/pokemon/Mewtwo/ed9eb0e7d3494c6992e06196f5b7cc05.svg
-> UNREADABLE
/Users/brandonreid/Science/pokemon/Bulbasaur/000007.gif -> UNREADABLE
1677  TOTAL IMAGES PROCESSED
```

[5]:
```
X = np.array(data).reshape(-1, IMG_SIZE, IMG_SIZE, 3) / 255.0 # Reshape and␣
 ↪scale images
y = to_categorical(labels, num_classes = len(pokemon)) # Labelize

print("[INFO] data matrix: {:.2f}MB".format(
        X.nbytes / (1024 * 1000.0)))
```

```
[INFO] data matrix: 362.23MB
```

[6]:
```
# Cross Validation
# Popular method to train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.2, random_state = SEED)
```

[7]:
```
# Defining ImageDataGenerator & construct data augmentation
# useful for fitting model to provide different image types, angles, rotations,␣
 ↪etc.
# I have excluded this due to how much time it adds to trail the modal
# after looking further https://github.com/keras-team/keras/issues/12683
# this is a known issue
aug = ImageDataGenerator(rotation_range = 25, # Degree range for random␣
 ↪rotations
                         width_shift_range = 0.1, # Range for horizontal shift
                         height_shift_range = 0.1, # Range for vertical shift
                         zoom_range = 0.2, # Range for random zoom
                         horizontal_flip = True, # Randomly flip inputs␣
 ↪horizontally
                         shear_range = 0.2, # Shear Intensity
                         fill_mode="nearest")

aug.fit(X_train) # ideally would augment training image set
```

[8]:
```
model = Sequential() # set up the modal for sequential layers

# SEQUENCE LAYER: CONV => RELU => POOL
model.add(Conv2D(32, (3, 3), padding="same", input_shape=IMAGE_DIMS)) #CONV
model.add(Activation("relu")) #RELU
model.add(BatchNormalization(axis=-1))
model.add(MaxPooling2D(pool_size=(3, 3))) # POOL
model.add(Dropout(0.25)) #Dropout to next sequence

# SEQUENCE LAYER: (CONV => RELU) * 2 => POOL
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=-1))

model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=1))

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25)) # Dropout to next sequence

# SEQUENCE LAYER: (CONV => RELU) * 2 => POOL
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=-1))

model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=-1))

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25)) # Dropout to next sequence

# SEQUENCE LAYER: first (and only) set of FC => RELU layers
model.add(Flatten())
model.add(Dense(1024))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5)) # Dropout to next sequence

# softmax classifier
model.add(Dense(len(pokemon))) # Dense is number of classes to predict
model.add(Activation("softmax"))

model.summary()

# optimizer
```

```
model.compile(Adam(lr=LEARN_RATE), loss='categorical_crossentropy',␣
 ↪metrics=["accuracy"])
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 96, 96, 32)        896
_________________________________________________________________
activation_1 (Activation)    (None, 96, 96, 32)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 96, 96, 32)        128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 64)        18496
_________________________________________________________________
activation_2 (Activation)    (None, 32, 32, 64)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 32, 32, 64)        256
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 32, 32, 64)        36928
_________________________________________________________________
activation_3 (Activation)    (None, 32, 32, 64)        0
_________________________________________________________________
batch_normalization_3 (Batch (None, 32, 32, 64)        128
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 128)       73856
_________________________________________________________________
activation_4 (Activation)    (None, 16, 16, 128)       0
_________________________________________________________________
batch_normalization_4 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 16, 16, 128)       147584
_________________________________________________________________
activation_5 (Activation)    (None, 16, 16, 128)       0
_________________________________________________________________
batch_normalization_5 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
```

```
dropout_3 (Dropout)          (None, 8, 8, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              8389632
_________________________________________________________________
activation_6 (Activation)    (None, 1024)              0
_________________________________________________________________
batch_normalization_6 (Batch (None, 1024)              4096
_________________________________________________________________
dropout_4 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 5125
_________________________________________________________________
activation_7 (Activation)    (None, 5)                 0
=================================================================
Total params: 8,678,149
Trainable params: 8,675,333
Non-trainable params: 2,816
_________________________________________________________________
```

[9]:
```
history = model.fit(X_train, y_train, batch_size=BS,
                    epochs=EPOCHS, verbose=1, validation_data=(X_test, y_test))

# history = model.fit_generator(aug.flow(X_train, y_train, batch_size=BS),
#                       validation_data=(X_test, y_test),
#                       steps_per_epoch=len(X_train) // BS,
#                       epochs=EPOCHS,
#                       verbose=1)
```

```
Train on 1341 samples, validate on 336 samples
Epoch 1/100
1341/1341 [==============================] - 50s 37ms/step - loss: 1.5503 -
accuracy: 0.5019 - val_loss: 1.5788 - val_accuracy: 0.3869
Epoch 2/100
1341/1341 [==============================] - 45s 34ms/step - loss: 0.8934 -
accuracy: 0.7278 - val_loss: 1.5542 - val_accuracy: 0.2351
Epoch 3/100
1341/1341 [==============================] - 46s 34ms/step - loss: 0.7302 -
accuracy: 0.7673 - val_loss: 1.6577 - val_accuracy: 0.1875
Epoch 4/100
1341/1341 [==============================] - 45s 33ms/step - loss: 0.6033 -
accuracy: 0.8061 - val_loss: 1.6550 - val_accuracy: 0.2411
Epoch 5/100
1341/1341 [==============================] - 45s 34ms/step - loss: 0.5659 -
accuracy: 0.8166 - val_loss: 2.0370 - val_accuracy: 0.1875
Epoch 6/100
1341/1341 [==============================] - 45s 34ms/step - loss: 0.4423 -
```

```
accuracy: 0.8553 - val_loss: 2.5136 - val_accuracy: 0.1994
Epoch 7/100
1341/1341 [==============================] - 45s 33ms/step - loss: 0.3935 -
accuracy: 0.8665 - val_loss: 2.7008 - val_accuracy: 0.2470
Epoch 8/100
1341/1341 [==============================] - 44s 33ms/step - loss: 0.3460 -
accuracy: 0.8792 - val_loss: 2.5530 - val_accuracy: 0.3036
Epoch 9/100
1341/1341 [==============================] - 47s 35ms/step - loss: 0.3364 -
accuracy: 0.8889 - val_loss: 2.1155 - val_accuracy: 0.4107
Epoch 10/100
1341/1341 [==============================] - 46s 35ms/step - loss: 0.3412 -
accuracy: 0.8852 - val_loss: 1.5435 - val_accuracy: 0.5655
Epoch 11/100
1341/1341 [==============================] - 45s 33ms/step - loss: 0.2893 -
accuracy: 0.9045 - val_loss: 1.3093 - val_accuracy: 0.6101
Epoch 12/100
1341/1341 [==============================] - 48s 36ms/step - loss: 0.2845 -
accuracy: 0.9008 - val_loss: 1.0530 - val_accuracy: 0.6905
Epoch 13/100
1341/1341 [==============================] - 48s 36ms/step - loss: 0.2791 -
accuracy: 0.8949 - val_loss: 0.8240 - val_accuracy: 0.7679
Epoch 14/100
1341/1341 [==============================] - 49s 36ms/step - loss: 0.2316 -
accuracy: 0.9113 - val_loss: 0.7726 - val_accuracy: 0.7827
Epoch 15/100
1341/1341 [==============================] - 49s 36ms/step - loss: 0.2086 -
accuracy: 0.9284 - val_loss: 0.7797 - val_accuracy: 0.7946
Epoch 16/100
1341/1341 [==============================] - 50s 37ms/step - loss: 0.2235 -
accuracy: 0.9239 - val_loss: 0.6989 - val_accuracy: 0.8155
Epoch 17/100
1341/1341 [==============================] - 50s 37ms/step - loss: 0.1965 -
accuracy: 0.9292 - val_loss: 0.7002 - val_accuracy: 0.8095
Epoch 18/100
1341/1341 [==============================] - 51s 38ms/step - loss: 0.2179 -
accuracy: 0.9195 - val_loss: 0.6816 - val_accuracy: 0.8036
Epoch 19/100
1341/1341 [==============================] - 50s 37ms/step - loss: 0.1692 -
accuracy: 0.9441 - val_loss: 0.6231 - val_accuracy: 0.8065
Epoch 20/100
1341/1341 [==============================] - 52s 39ms/step - loss: 0.1628 -
accuracy: 0.9508 - val_loss: 0.8104 - val_accuracy: 0.7946
Epoch 21/100
1341/1341 [==============================] - 51s 38ms/step - loss: 0.1800 -
accuracy: 0.9262 - val_loss: 0.6092 - val_accuracy: 0.8244
Epoch 22/100
1341/1341 [==============================] - 51s 38ms/step - loss: 0.1489 -
```

```
accuracy: 0.9508 - val_loss: 0.7116 - val_accuracy: 0.8036
Epoch 23/100
1341/1341 [==============================] - 51s 38ms/step - loss: 0.1712 -
accuracy: 0.9396 - val_loss: 0.6660 - val_accuracy: 0.8244
Epoch 24/100
1341/1341 [==============================] - 51s 38ms/step - loss: 0.1150 -
accuracy: 0.9575 - val_loss: 0.5849 - val_accuracy: 0.8214
Epoch 25/100
1341/1341 [==============================] - 51s 38ms/step - loss: 0.1136 -
accuracy: 0.9597 - val_loss: 0.5881 - val_accuracy: 0.8274
Epoch 26/100
1341/1341 [==============================] - 51s 38ms/step - loss: 0.1205 -
accuracy: 0.9575 - val_loss: 0.5462 - val_accuracy: 0.8393
Epoch 27/100
1341/1341 [==============================] - 53s 40ms/step - loss: 0.1470 -
accuracy: 0.9567 - val_loss: 0.6029 - val_accuracy: 0.8423
Epoch 28/100
1341/1341 [==============================] - 52s 39ms/step - loss: 0.1203 -
accuracy: 0.9545 - val_loss: 0.5968 - val_accuracy: 0.8333
Epoch 29/100
1341/1341 [==============================] - 53s 40ms/step - loss: 0.1324 -
accuracy: 0.9523 - val_loss: 0.6392 - val_accuracy: 0.8482
Epoch 30/100
1341/1341 [==============================] - 52s 38ms/step - loss: 0.1193 -
accuracy: 0.9553 - val_loss: 0.6015 - val_accuracy: 0.8423
Epoch 31/100
1341/1341 [==============================] - 52s 39ms/step - loss: 0.1140 -
accuracy: 0.9605 - val_loss: 0.6175 - val_accuracy: 0.8423
Epoch 32/100
1341/1341 [==============================] - 52s 39ms/step - loss: 0.1176 -
accuracy: 0.9560 - val_loss: 0.7695 - val_accuracy: 0.8274
Epoch 33/100
1341/1341 [==============================] - 52s 39ms/step - loss: 0.1106 -
accuracy: 0.9590 - val_loss: 0.6264 - val_accuracy: 0.8423
Epoch 34/100
1341/1341 [==============================] - 52s 39ms/step - loss: 0.1038 -
accuracy: 0.9687 - val_loss: 0.6512 - val_accuracy: 0.8452
Epoch 35/100
1341/1341 [==============================] - 51s 38ms/step - loss: 0.1052 -
accuracy: 0.9620 - val_loss: 0.6187 - val_accuracy: 0.8482
Epoch 36/100
1341/1341 [==============================] - 51s 38ms/step - loss: 0.1085 -
accuracy: 0.9627 - val_loss: 0.6091 - val_accuracy: 0.8423
Epoch 37/100
1341/1341 [==============================] - 52s 39ms/step - loss: 0.1002 -
accuracy: 0.9709 - val_loss: 0.6233 - val_accuracy: 0.8452
Epoch 38/100
1341/1341 [==============================] - 51s 38ms/step - loss: 0.1053 -
```

```
accuracy: 0.9672 - val_loss: 0.6216 - val_accuracy: 0.8512
Epoch 39/100
1341/1341 [==============================] - 52s 39ms/step - loss: 0.1078 -
accuracy: 0.9582 - val_loss: 0.7105 - val_accuracy: 0.8244
Epoch 40/100
1341/1341 [==============================] - 50s 37ms/step - loss: 0.0949 -
accuracy: 0.9687 - val_loss: 0.5789 - val_accuracy: 0.8423
Epoch 41/100
1341/1341 [==============================] - 48s 36ms/step - loss: 0.0712 -
accuracy: 0.9769 - val_loss: 0.6102 - val_accuracy: 0.8363
Epoch 42/100
1341/1341 [==============================] - 51s 38ms/step - loss: 0.0977 -
accuracy: 0.9717 - val_loss: 0.5599 - val_accuracy: 0.8512
Epoch 43/100
1341/1341 [==============================] - 54s 40ms/step - loss: 0.0847 -
accuracy: 0.9746 - val_loss: 0.5439 - val_accuracy: 0.8601
Epoch 44/100
1341/1341 [==============================] - 52s 39ms/step - loss: 0.0788 -
accuracy: 0.9724 - val_loss: 0.5344 - val_accuracy: 0.8512
Epoch 45/100
1341/1341 [==============================] - 49s 37ms/step - loss: 0.0923 -
accuracy: 0.9717 - val_loss: 0.5910 - val_accuracy: 0.8482
Epoch 46/100
1341/1341 [==============================] - 47s 35ms/step - loss: 0.0901 -
accuracy: 0.9732 - val_loss: 0.5803 - val_accuracy: 0.8423
Epoch 47/100
1341/1341 [==============================] - 44s 33ms/step - loss: 0.1081 -
accuracy: 0.9620 - val_loss: 0.6365 - val_accuracy: 0.8512
Epoch 48/100
1341/1341 [==============================] - 46s 34ms/step - loss: 0.0662 -
accuracy: 0.9761 - val_loss: 0.5934 - val_accuracy: 0.8512
Epoch 49/100
1341/1341 [==============================] - 46s 34ms/step - loss: 0.0865 -
accuracy: 0.9672 - val_loss: 0.5836 - val_accuracy: 0.8512
Epoch 50/100
1341/1341 [==============================] - 47s 35ms/step - loss: 0.0762 -
accuracy: 0.9739 - val_loss: 0.5401 - val_accuracy: 0.8571
Epoch 51/100
1341/1341 [==============================] - 51s 38ms/step - loss: 0.0734 -
accuracy: 0.9769 - val_loss: 0.5372 - val_accuracy: 0.8601
Epoch 52/100
1341/1341 [==============================] - 48s 36ms/step - loss: 0.0840 -
accuracy: 0.9761 - val_loss: 0.6061 - val_accuracy: 0.8542
Epoch 53/100
1341/1341 [==============================] - 47s 35ms/step - loss: 0.0853 -
accuracy: 0.9717 - val_loss: 0.5482 - val_accuracy: 0.8631
Epoch 54/100
1341/1341 [==============================] - 47s 35ms/step - loss: 0.0701 -
```

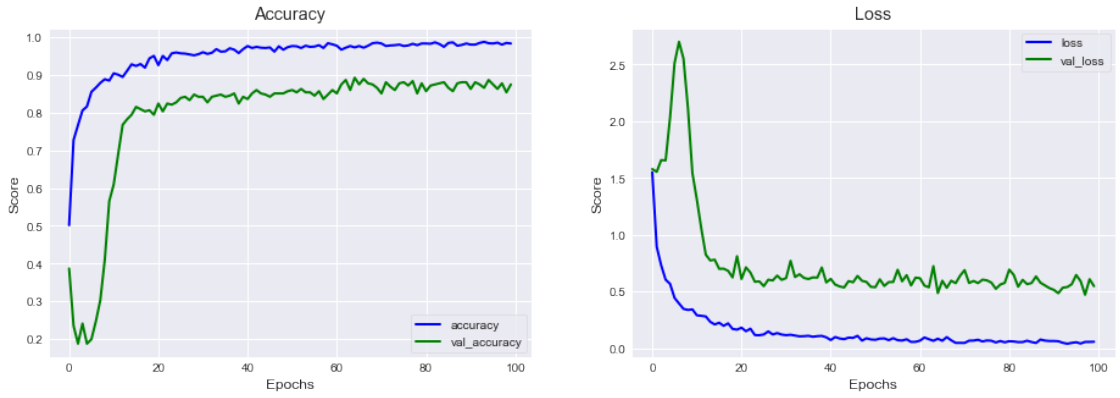
```
accuracy: 0.9776 - val_loss: 0.5821 - val_accuracy: 0.8542
Epoch 55/100
1341/1341 [==============================] - 47s 35ms/step - loss: 0.0875 -
accuracy: 0.9746 - val_loss: 0.5832 - val_accuracy: 0.8542
Epoch 56/100
1341/1341 [==============================] - 48s 36ms/step - loss: 0.0700 -
accuracy: 0.9754 - val_loss: 0.6908 - val_accuracy: 0.8452
Epoch 57/100
1341/1341 [==============================] - 44s 33ms/step - loss: 0.0667 -
accuracy: 0.9791 - val_loss: 0.5891 - val_accuracy: 0.8571
Epoch 58/100
1341/1341 [==============================] - 46s 35ms/step - loss: 0.0778 -
accuracy: 0.9717 - val_loss: 0.6435 - val_accuracy: 0.8363
Epoch 59/100
1341/1341 [==============================] - 49s 36ms/step - loss: 0.0545 -
accuracy: 0.9843 - val_loss: 0.5528 - val_accuracy: 0.8482
Epoch 60/100
1341/1341 [==============================] - 46s 34ms/step - loss: 0.0552 -
accuracy: 0.9814 - val_loss: 0.6210 - val_accuracy: 0.8601
Epoch 61/100
1341/1341 [==============================] - 45s 34ms/step - loss: 0.0658 -
accuracy: 0.9776 - val_loss: 0.6162 - val_accuracy: 0.8512
Epoch 62/100
1341/1341 [==============================] - 45s 34ms/step - loss: 0.0934 -
accuracy: 0.9672 - val_loss: 0.5510 - val_accuracy: 0.8750
Epoch 63/100
1341/1341 [==============================] - 46s 35ms/step - loss: 0.0785 -
accuracy: 0.9724 - val_loss: 0.5361 - val_accuracy: 0.8869
Epoch 64/100
1341/1341 [==============================] - 44s 33ms/step - loss: 0.0643 -
accuracy: 0.9769 - val_loss: 0.7227 - val_accuracy: 0.8601
Epoch 65/100
1341/1341 [==============================] - 44s 33ms/step - loss: 0.0828 -
accuracy: 0.9732 - val_loss: 0.4851 - val_accuracy: 0.8929
Epoch 66/100
1341/1341 [==============================] - 45s 34ms/step - loss: 0.0649 -
accuracy: 0.9769 - val_loss: 0.5938 - val_accuracy: 0.8750
Epoch 67/100
1341/1341 [==============================] - 44s 33ms/step - loss: 0.0968 -
accuracy: 0.9724 - val_loss: 0.5318 - val_accuracy: 0.8899
Epoch 68/100
1341/1341 [==============================] - 45s 34ms/step - loss: 0.0688 -
accuracy: 0.9776 - val_loss: 0.5941 - val_accuracy: 0.8780
Epoch 69/100
1341/1341 [==============================] - 44s 33ms/step - loss: 0.0459 -
accuracy: 0.9843 - val_loss: 0.5724 - val_accuracy: 0.8750
Epoch 70/100
1341/1341 [==============================] - 45s 34ms/step - loss: 0.0460 -
```

```
accuracy: 0.9858 - val_loss: 0.6388 - val_accuracy: 0.8661
Epoch 71/100
1341/1341 [==============================] - 44s 33ms/step - loss: 0.0455 -
accuracy: 0.9836 - val_loss: 0.6877 - val_accuracy: 0.8512
Epoch 72/100
1341/1341 [==============================] - 44s 33ms/step - loss: 0.0655 -
accuracy: 0.9769 - val_loss: 0.5737 - val_accuracy: 0.8810
Epoch 73/100
1341/1341 [==============================] - 46s 34ms/step - loss: 0.0665 -
accuracy: 0.9784 - val_loss: 0.5921 - val_accuracy: 0.8690
Epoch 74/100
1341/1341 [==============================] - 45s 34ms/step - loss: 0.0734 -
accuracy: 0.9791 - val_loss: 0.5728 - val_accuracy: 0.8601
Epoch 75/100
1341/1341 [==============================] - 45s 33ms/step - loss: 0.0587 -
accuracy: 0.9806 - val_loss: 0.6036 - val_accuracy: 0.8780
Epoch 76/100
1341/1341 [==============================] - 44s 33ms/step - loss: 0.0675 -
accuracy: 0.9769 - val_loss: 0.5974 - val_accuracy: 0.8810
Epoch 77/100
1341/1341 [==============================] - 44s 33ms/step - loss: 0.0648 -
accuracy: 0.9784 - val_loss: 0.5772 - val_accuracy: 0.8720
Epoch 78/100
1341/1341 [==============================] - 47s 35ms/step - loss: 0.0498 -
accuracy: 0.9828 - val_loss: 0.5228 - val_accuracy: 0.8839
Epoch 79/100
1341/1341 [==============================] - 45s 33ms/step - loss: 0.0625 -
accuracy: 0.9791 - val_loss: 0.5604 - val_accuracy: 0.8512
Epoch 80/100
1341/1341 [==============================] - 48s 36ms/step - loss: 0.0510 -
accuracy: 0.9836 - val_loss: 0.5755 - val_accuracy: 0.8780
Epoch 81/100
1341/1341 [==============================] - 46s 34ms/step - loss: 0.0609 -
accuracy: 0.9836 - val_loss: 0.6922 - val_accuracy: 0.8571
Epoch 82/100
1341/1341 [==============================] - 46s 34ms/step - loss: 0.0585 -
accuracy: 0.9828 - val_loss: 0.6459 - val_accuracy: 0.8720
Epoch 83/100
1341/1341 [==============================] - 46s 34ms/step - loss: 0.0529 -
accuracy: 0.9866 - val_loss: 0.5417 - val_accuracy: 0.8750
Epoch 84/100
1341/1341 [==============================] - 46s 34ms/step - loss: 0.0531 -
accuracy: 0.9821 - val_loss: 0.6018 - val_accuracy: 0.8780
Epoch 85/100
1341/1341 [==============================] - 46s 35ms/step - loss: 0.0653 -
accuracy: 0.9746 - val_loss: 0.5642 - val_accuracy: 0.8810
Epoch 86/100
1341/1341 [==============================] - 45s 33ms/step - loss: 0.0535 -
```

```
accuracy: 0.9851 - val_loss: 0.5734 - val_accuracy: 0.8661
Epoch 87/100
1341/1341 [==============================] - 46s 34ms/step - loss: 0.0449 -
accuracy: 0.9866 - val_loss: 0.6318 - val_accuracy: 0.8571
Epoch 88/100
1341/1341 [==============================] - 48s 36ms/step - loss: 0.0754 -
accuracy: 0.9776 - val_loss: 0.5764 - val_accuracy: 0.8780
Epoch 89/100
1341/1341 [==============================] - 47s 35ms/step - loss: 0.0662 -
accuracy: 0.9799 - val_loss: 0.5530 - val_accuracy: 0.8810
Epoch 90/100
1341/1341 [==============================] - 46s 34ms/step - loss: 0.0626 -
accuracy: 0.9836 - val_loss: 0.5291 - val_accuracy: 0.8810
Epoch 91/100
1341/1341 [==============================] - 45s 34ms/step - loss: 0.0628 -
accuracy: 0.9806 - val_loss: 0.5133 - val_accuracy: 0.8631
Epoch 92/100
1341/1341 [==============================] - 45s 34ms/step - loss: 0.0606 -
accuracy: 0.9806 - val_loss: 0.4840 - val_accuracy: 0.8810
Epoch 93/100
1341/1341 [==============================] - 46s 34ms/step - loss: 0.0467 -
accuracy: 0.9851 - val_loss: 0.5326 - val_accuracy: 0.8750
Epoch 94/100
1341/1341 [==============================] - 45s 34ms/step - loss: 0.0386 -
accuracy: 0.9881 - val_loss: 0.5385 - val_accuracy: 0.8661
Epoch 95/100
1341/1341 [==============================] - 45s 33ms/step - loss: 0.0459 -
accuracy: 0.9843 - val_loss: 0.5629 - val_accuracy: 0.8869
Epoch 96/100
1341/1341 [==============================] - 44s 33ms/step - loss: 0.0517 -
accuracy: 0.9836 - val_loss: 0.6448 - val_accuracy: 0.8750
Epoch 97/100
1341/1341 [==============================] - 46s 35ms/step - loss: 0.0402 -
accuracy: 0.9858 - val_loss: 0.5903 - val_accuracy: 0.8631
Epoch 98/100
1341/1341 [==============================] - 46s 34ms/step - loss: 0.0544 -
accuracy: 0.9806 - val_loss: 0.4704 - val_accuracy: 0.8780
Epoch 99/100
1341/1341 [==============================] - 47s 35ms/step - loss: 0.0546 -
accuracy: 0.9851 - val_loss: 0.6073 - val_accuracy: 0.8542
Epoch 100/100
1341/1341 [==============================] - 46s 34ms/step - loss: 0.0561 -
accuracy: 0.9836 - val_loss: 0.5463 - val_accuracy: 0.8750
```

[10]:
```
# Plot learning curves
train_loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
train_acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

acc_df = pd.Series(train_acc)
val_acc_df = pd.Series(val_acc)
loss_df = pd.Series(train_loss)
val_loss_df = pd.Series(val_loss)

fig = plt.figure(figsize = (14, 5))

plt.subplot(121)
sns.lineplot(data=acc_df, color='blue', label="accuracy", linewidth=2).
 ↪set_title("Accuracy")
sns.lineplot(data=val_acc_df, color='green', label="val_accuracy", linewidth=2).
 ↪set(xlabel='Epochs', ylabel='Score')
plt.legend(loc='lower right')

plt.subplot(122)
sns.lineplot(data=loss_df, color='blue', label="loss", linewidth=2).
 ↪set_title("Loss")
sns.lineplot(data=val_loss_df, color='green', label="val_loss", linewidth=2).
 ↪set(xlabel='Epochs', ylabel='Score')
plt.legend(loc='upper right')
plt.show()
```

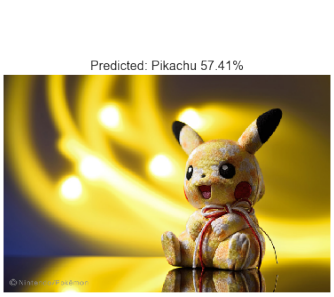


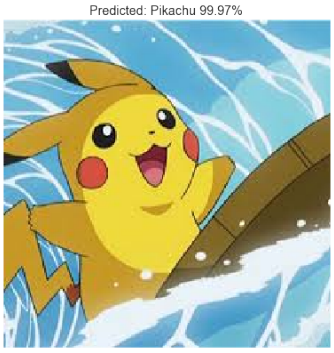
[11]:
```
# return image as array from web using requests
def get_image(url):
    r = requests.get(url, stream = True).raw
    return np.asarray(bytearray(r.read()), dtype="uint8")
```

[66]:
```
# prediction and plot function for image
def predict_image(url, row, pos):
```

```
    # process image
    image = get_image(url)
    image = cv.imdecode(image, cv.IMREAD_COLOR)
    original_img = image
    image = cv.resize(image, (IMG_SIZE, IMG_SIZE)) # Resizing image to (96, 96)
    image = image.reshape(-1, IMG_SIZE, IMG_SIZE, 3) / 255.0 # Reshape and␣
↪scale resized image

    # run prediction
    preds = model.predict(image) # Predicting image
    pred_class = np.argmax(preds) # Defining predicted class

    # plot image with prediction outcome
    plots.add_subplot(row, 3, pos)
    plt.imshow(original_img[:, :, ::-1])
    plt.title(f'Predicted: {pokemon[pred_class]} {round(preds[0][pred_class] *␣
↪100, 2)}%')
    plt.axis('off')
```

[102]:
```
# predict pikachu images from web
pikachu3D = 'https://www.nme.com/wp-content/uploads/2019/05/
↪MV5BZTNmOGE1ZmQtYjZmYy00MDhkLThkZjYtNzkyZDYxNTA5ZTE0XkEyXkFqcGdeQXVyNjg2NjQwMDQ@.
↪jpg'
pikachuToy = 'https://soranews24.com/wp-content/uploads/sites/3/2019/11/pk-3.
↪png?w=640&h=426'
pikachuOG = 'https://encrypted-tbn0.gstatic.com/images?
↪q=tbn%3AANd9GcQxX6o-s7rgiQGUrqeJJCws4uPLz9AxUqVMcA&usqp=CAU'

plots = plt.figure(figsize = (20, 20))

predict_image(pikachu3D, 1, 1)
predict_image(pikachuToy, 1, 2)
predict_image(pikachuOG, 1, 3)

plt.show()
```

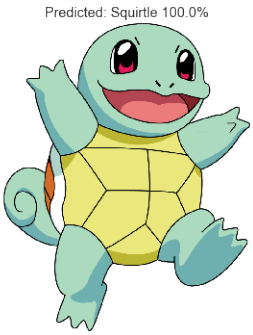




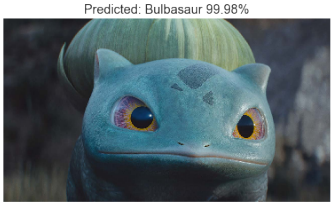
[103]:
```
# predict mewtwo images from web
mewtwo3D = 'https://cdn.now.howstuffworks.com/media-content/
 ↪3beb4197-0e43-4943-95b8-c432f6b0071a-1920-1080.jpg'
mewtwoToy = 'https://target.scene7.com/is/image/Target/
 ↪GUEST_e5bb766e-140c-48ad-a323-1fde69087fed?wid=488&hei=488&fmt=pjpeg'
mewtwoOG = 'https://m.media-amazon.com/images/M/
 ↪MV5BNjk1NTc3MDEtZDM4ZS00YzE2LThhNjEtMzdmZDcwMGJiNTA5XkEyXkFqcGdeQW1yb3NzZXI@.
 ↪_V1_UX477_CR0,0,477,268_AL_.jpg'

plots = plt.figure(figsize = (20, 20))

predict_image(mewtwo3D, 2, 1)
predict_image(mewtwoToy, 2, 2)
predict_image(mewtwoOG, 2, 3)

plt.show()
```

[104]:
```
# predict squirtle images from web
squirtle3D = 'https://66.media.tumblr.com/6ac780a38a8d9eafed01daf41d6cd45b/
 ↪tumblr_ps1sykZPEV1vplr8j_400.png'
squirtleToy = 'https://ae01.alicdn.com/kf/Hf096a66261864013886bd82540d9f4757/
 ↪GK-Anime-Bulbasaur-Squirtle-Charmander-pikaqu-Charizard-Resin-Statue-Figure-Small-Fire-Drag
 ↪jpg'
squirtleOG = 'https://aguycalledkane.files.wordpress.com/2017/04/squirtle.png?
 ↪w=840'

plots = plt.figure(figsize = (20, 20))

predict_image(squirtle3D, 3, 1)
predict_image(squirtleToy, 3, 2)
predict_image(squirtleOG, 3, 3)
```

```
plt.show()

# Failure with Toy image: Predicted was Bulbasaur, most likely do to edges in␣
 ↪image
```



[105]:
```
# predict bulbasuar images from web
bulbasuar3D = 'https://cdn1.thr.com/sites/default/files/2019/05/
 ↪detective_pikachu-bulbasaur-publicity-h_2019.jpg'
bulbasuarOG = 'https://i.pinimg.com/736x/6f/aa/74/
 ↪6faa74e32ce5f00f2dbc9c0b095cda52.jpg'
bulbasuarToy = 'https://www.choozily.com/wp-content/uploads/
 ↪Funko-Pop-Bulbasaur-Pokemon-Vinyl-Figure-Choozily.jpg'

plots = plt.figure(figsize = (20, 20))

predict_image(bulbasuar3D, 1, 1)
predict_image(bulbasuarToy, 1, 2)
predict_image(bulbasuarOG, 1, 3)

plt.show()
```



[107]:
```
# predict charmander images from web
charmanderToy = 'https://ak1.ostkcdn.com/images/products/is/images/direct/
 ↪2c9a494898df8ce9f85130f15683fdd2d80b9c51/
 ↪Warp-Gadgets-Pok%C3%A9mon-Bundle-Funko-Pop-Charmander-and-Bulbasaur-2-Items.
 ↪jpg'
charmander3D = 'https://i.pinimg.com/474x/48/8f/c4/
 ↪488fc45206b15c68313706ca638d9441.jpg'
charmanderOG = 'https://i.ebayimg.com/images/g/-Z4AAOSwkShY8ocn/s-l400.jpg'

plots = plt.figure(figsize = (20, 20))

predict_image(charmander3D, 1, 1)
predict_image(charmanderToy, 1, 2)
predict_image(charmanderOG, 1, 3)

plt.show()

# Failure: 3D image from detective pikachu movie, Predicted Pikachu
```

[ ]:
```
## References
# https://medium.com/@kelfun5354/
 ↪building-a-simple-pokemon-convolutional-neural-net-cc724a8fb47d
# https://www.pyimagesearch.com/2018/04/16/
 ↪keras-and-convolutional-neural-networks-cnns/
# https://www.youtube.com/watch?v=FmpDIaiMIeA
# https://www.geeksforgeeks.org/image-classifier-using-cnn/
# https://www.pyimagesearch.com/2019/07/08/
 ↪keras-imagedatagenerator-and-data-augmentation/

## TODO
```

```
# Data/Image augmentation optimization
# crawl for more images on 3D versions
# Provide callback to fit model to monitor best fits
```