P&S Module 2019 Report Pokédex

Dejan Bozin, Florian Bolli
D-ITET, ETH Zürich
{bozind, bollif}@student.ethz.ch

1. Introduction

At the beginning of this module students were instructed into Basics of Deep Learning and Computer Vision. Students were asked to come up with an idea for the project and realize it with the previous learned tools of Deep Learning and Computer Vision. Since students still have the opportunity to experiment and work on an arbitrary chosen project, we decided to do something uncommon and special. Thus, creating a device, that will be able to recognize Pokémon characters would be a lot of fun and could be an interesting toy for children in the future. This device already exists, but only in the corresponding TV series, known as Pokédex. Realizing such a device made our childhood-dream come true. We focused especially on eleven creatures of the first generation. For this purpose, we had to contact an Anime store to help us providing fluffy Pokémon to generate our Dataset. The objective in classification is to find appropriate labels for unlabeled data using a deep learning model. To realize our model the most effective option for image processing is building a convolutional neural network (CNNs). Even for simple CNNs the model gives a good classification accuracy. More details about our achieved accuracy is mentioned in Section Results. Another goal of our project was to keep the use the device very simple and build it as similar to the original fictive one as possible. With the portable Raspberry Pi and its Camera, it is possible to capture photos everywhere. After capturing the Pokémon, it will output its properties acoustically. Further information of Methods will be provided in the Section Methods. A detailed description of our approach for collecting data as well as deeper view into our data processing part will be given there. In Addition, there will be a brief introduction of concepts of CNNs and other necessary parts of Machine Learning theory.

2. Methods

2.1. Data collection

Since Pokémon does not occur in our nature, we had to consider an alternative for the data collection. As a solution for this problem we found an anime shop, which is in possession of different Pokémon plush figures. The owner was so kind that he made it possible for us to take pictures of the figures and so we could create our dataset. We have focused on a total of eleven figures. In the table below, you can find the different Pokémon with a photo example and corresponding names. From each Pokémon 1500 photos were taken and used for the training of our model. For a fast collection we used the Burst-Mode of our smartphones to capture all pictures as fast as possible. Attention was paid to have same background for whole training data. Unlike for our test data, this consists of Pokémon, which were placed in the shop in front of various backgrounds (see Figure 2). Lowering computational expense is required, since our images have a too high resolution. An appropriate choice for resizing was a resolution of 100 by 75.

2.1.1 Collected train data

The following pictures are examples of our train data set. As already mentioned, there are about 1500 of these pictures for each class.

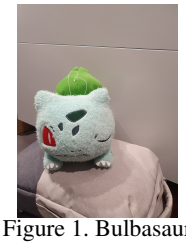
Figure 1. Bulbasaur

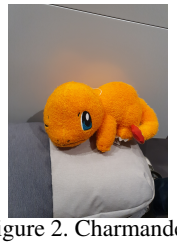
Figure 2. Charmander


Figure 3. Dratini

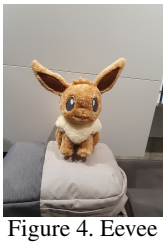
Figure 4. Eevee


Figure 5. Jigglypuff


Figure 6. Jolteon


Figure 7. Meowth


Figure 8. Pichu


Figure 9. Pikatchu


Figure 10. Squirtle


Figure 11. Vaporeon

2.1.2 Collected test data

We also took the photos for our test data in the Manga shop. To show, how their background and lightning differs, we will show some test images of Eevee:


Figure 12. Eevee Test 1


Figure 13. Eevee Test 2

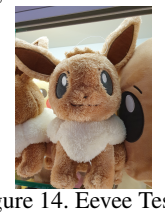
Figure 14. Eevee Test 3


Figure 15. Eevee Test 4

2.2. Model

In session 03, we have leraned a lot about convolution networks. Since we are categorizing classes visually, a convolution network is the best approach. We chose a model similar to an example we found on the internet (blog.keras.io). It was originally made for recognizing only cats and dogs. We tried to tune it a bit since we have 11 classes and an image size of 100 x 75 pixels. But it worked best as it was proposed on the website. It is now a network with: The input layer of the size of the image (100 x 75 = 7500 neurons) Two hidden layers which make a convolution with kernel sizes of 7x7 respectively 5x5 pixels and Relu as activation function The output layer with 11 neorons and Softmax activation. For more details see the declaration of the model in the code.

```
# define a model
num_train_samples = 15000
num_test_samples = 50
input_shape = (75,100,3)

kernel_sizes = [(7, 7), (5, 5)]
num_kernels = [20, 25]

pool_sizes = [(2, 2), (2, 2)]
pool_strides = [(2, 2), (2, 2)]

num_hidden_units = 200

x = Input(shape=input_shape)
y = Conv2D(num_kernels[0], kernel_sizes[0], activation='relu')(x)
y = MaxPooling2D(pool_sizes[0], pool_strides[0])(y)
y = Conv2D(num_kernels[1], kernel_sizes[1], activation='relu')(y)
y = MaxPooling2D(pool_sizes[1], pool_strides[1])(y)
y = Flatten()(y)
y = Dense(num_hidden_units, activation='relu')(y)
y = Dense(num_classes, activation='softmax')(y)
model = Model(x, y)
```

Figure 16. Definition of the model

2.3. Training

Even if the program had to work on the raspberry pi it was not such a good idea to train the network on the little computer. It would be possible indeed, but it would take several days of training to get a satisfying result. I decided to train the network on a laptop for 10 epochs, with a batch size of 16. More than 10 epochs did not improve our results because with the train data there was already an accuracy of 99.99It took approximately one hour to train the network like this. Then we copied the weight file to the raspberry and disabled the training function for further actions.

2.4. Single classification

We let the Keras framework do what it can best: recognize some well structured data sets. But that was not very useful yet. The Pokédex would only really work, if we had the possibility for live recognition. We firstly had to make pictures with the raspberry, and then we had to figure out, how we can present our network one single (freshly

made) picture to recognize. We finally found out that the method: model.predict(imagename) can not only recognize whole data sets, but also single images. But the output of the predict() function was not that readable. We just got the results of the output layer neurons. With the argmax() function, we found out, which Pokémon-Id was predicted and translated the id to the actual class name.

2.5. Special Effects

We finally had a working Pokémon recognizing machine. But it was still not that useful because it was not portable at all. We had the idea to make it wireless, since weve got this possibility thanks to the size of the raspberry pi. We connected it to a mobile hotspot and configured an SSH connection, executable from the mobile phone. Also we had to power it with a power bank. We wanted it to work without any screen, so it had to tell us the results acoustically (like in the movies). We therefore collected some information about each Pokémon on a website (bulbapedia.bulbagarden.net). After writing a little summary for each class, we transformed the written information to an mp3 file containing a spoken text, using an online text to speech converter. Those texts can be found in the appendix. Nevertheless we could not demonstrate it without screen, because it was too difficult to aim at a fluffy Pokémon with the camera.

2.6. Other Problems

Compiling the model worked pretty well at the beginning. But there were some problems, that slew us down a little bit. The first one occured as we tried, classifying single images. It took us a long time to recognize the mistake, that we have to edit our single image exactly the same way as the training data. In this case, we forgot that we have converted the colors of our training data from an int between 0 and 255 into a float from 0 to 1. Since we have not made this with our single image, the result was pretty random. Another problem was the recognition of keyboard inputs. We wanted the Raspberry Pi to take a snapshot and recognize the Pokémon, every time a specific key is pressed. Of course there were some python libraries that had this functions, but they all required root access. We could not start the python script with "Sudo" because appearently all the libraries were not installed. We tried to install them, but this ended up in resetting the whole SD card. We finally solved it through the command line. We just cought an empty input (Just an enter keypress) of the command line. Like this, we could also control it via SSH in a simple way.

3. Experiment Results

The two collected datasets were utilized to train and validate images and analyse the accuracy by comparing the output with the testing data. Since the testing data were the same fluffy Pokémon as used for training but only with different background, we expected to have a quite high accuracy. For Pokémon illustrated on Images in the Internet, the classifier does not work very well. But for our testing data (and similar photos of the fluffy Pokémon, we achieved an accuracy of 85 percent. Unfortunately, the device doesnt work well for rotated captured pictures because we trained it only with upright token pictures. This few things could be improved, for example with letting the datagenerator also rotating images. We think it was a pitty, that the Pokédex in the end did not work without a screen. We underestimated the difficulty to capture a photo blindly. Nevertheless we could present a working prototype of a real Pokédex, with a quite satisfying performance. We are supprised, that even very similar Pokémon (like Pichu, Pikatchu or Jolteon) could be very well differed by our network.

4. Contributions

Since we have had a fixed time and place for working on this project, we did nearly everything together. There were some processes, which could be executed parallelly to be more productive yet. It was not 100% divided up, but we shared the work more of less like this:

4.1. Together

- Organization of the project
- Datacollection
- Planned a model from session3
- Make the model work
- Run the code on the Raspberry Pi

4.2. Dejan

- Software for rescalling and renaming images
- Creating .mp3 files for the spoken results
- let the Raspberry Pi take pictures for live recognition
- Power Point Presentation for the Prototype
- Report: Introduction, Data collection, Results

4.3. Florian

- Optimizing and training the model
- Optimizing the application as whole (Portable, make the little details work)
- Report: Methods, Contributions, format in Latex

References

PnS Module 2019. *Deeplearning on Raspberry Pi*, https://pns2019.github.io/

The Keras Blog. *Building powerful image classification models using very little data*, https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

The Keras Documentation. *Keras: The Python Deep Learning library*, https://keras.io/

5. Appendix

The following texts are short descriptions of our classes. The Raspberry Pi will tell those sentenses, if it recognizes a Pokémon.

5.1. Bulbasaur

This is a Bulbasaur, Pokémon number 001. Bulbasaur is a small, quadruped Pokémon that has blue-green skin with darker patches. It can grow up to 0.7m and 6.9 kg.

5.2. Charmander

This is a Charmander, Pokémon number 004. It can grow up to 0.6m and 8.5kg. Charmander is a bipedal, reptilian Pokémon with a primarily orange body and blue eyes.

5.3. Dratini

This is a Dratini, Pokémon number 147. Dratini is a serpentine Pokémon with a blue body and a white underside. It can grow up to 1.8m and 3.3kg.

5.4. Eevee

This is an Eevee, Pokémon number 133. Eevee is a mammalian, quadruped Pokémon with primarily brown fur. It can grow up to 0.3m and 6.5kg.

5.5. Jigglypuff

This is a Jigglypuff, Pokémon number 039. Jigglypuff a pink Pokémon with a spherical body. It can grow up to 0.5m and 5.5kg.

5.6. Jolteon

This is a Jolten, Pokémon number 135. Jolteon is a quadruped, mammalian Pokémon. It is covered in yellow fur with a spiky fringe around its tail and a white ruff around its neck. It can grow up to 0.8m and 24.5kg.

5.7. Meowth

This is a Meowth, Pokémon number 52. Meowth is a small, feline Pokémon with cream-colored fur that turns brown at the tips of its hind paws and tail. It can grow up to 0.4m and 4.2kg.

5.8. Pichu

This is a Pichu, Pokémon number 172. Pichu is a small, ground-dwelling rodent Pokémon with pale yellow fur. It can grow up to 0.3m and 2.0kg.

5.9. Pikatchu

This is a Pikatchu, Pokémon number 025. Pikachu is a short, chubby rodent Pokémon. It is covered in yellow fur with two horizontal brown stripes on its back. It can grow up to 0.4m and 6kg.

5.10. Squirtle

This is a Squirtle, Pokémon number 007. Squirtle is a small Pokémon that resembles a light blue turtle. It can grow up to 0.5m and 9kg.

5.11. Vaporeon

This is a Vaporeon, Pokémon number 134. Vaporeon is a Pokémon that shares physical traits with both aquatic and land animals. It can grow up to 1m and 29kg.