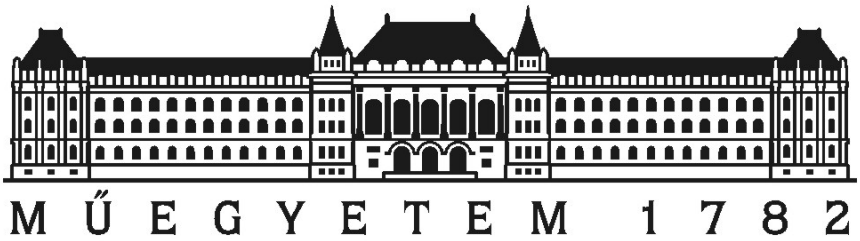


Media Informatic Systems

Image recognition task

Progress report

Author
Carlos Sánchez Páez

Budapest University of Technology and Economics

Academic year 2018-2019

Contents

List of Figures **1**

1 Description of the task **2**

2 Dataset **2**

3 Diary of work **2**

4 Preprocessing of the images **3**

5 Structure of the neural net **3**

6 The current state of the model **4**

7 The future of the project **5**

8 Examples **5**

9 Annex: classes contained in the dataset **7**

List of Figures

1 Loading the CIFAR100 dataset 2
2 Preprocessing of the images. 3
3 Accuracy and loss of the model 4
4 Bee prediction 5
5 Flipped bee prediction 6
6 Porcupine prediction 6
7 Bear prediction 7

1 Description of the task

My task consists on the development of a neural net which will be able to classify different types of RGB images according to the used dataset.

2 Dataset

The train and test images will be taken from *CIFAR100*. It is a dataset which contains 50,000 32x32 color training images, labeled over 100 categories, and 10,000 test images.

The main advantage of this dataset is that it is included in the *keras* dataset. This allows us to load the data by just executing this sentences:

```
from keras.datasets import cifar100
(x_train, y_train), (x_test, y_test) = cifar100.load_data()
```

Figure 1: Loading the CIFAR100 dataset

3 Diary of work

1. **Familiarize with *TensorFlow***. The first thing I did was reading some *TensorFlow* tutorials. Specifically, I developed a classifier which was able to distinguish between dogs and cats with the help of a tutorial.

2. **Try to adapt the algorithm**. I thought about converting the dogs and cats classifier into a biger one, so I made the corresponding changes to use it with the *Caltech-101* dataset. The main problem was that the number of images was too low for some categories. That is why I only achieved a 34.5% success rate.

3. **Jump to Caltech-256**. As there were too few images in Caltech-101, I downloaded the Caltech-256 dataset and started to work with it. But I had the opposite problem: as there were too much images, my local PC memory was not enough. To fix this, I created a virtual machine using Google Cloud (30GB of RAM). After 8 hours of training, the accuracy that I achieved was 0.39%. That is because the images for the categories were too similar and I did not apply any preprocessing.

4. **Decide to work with *Caltech-101* and local PC**. As the results obtained in *Caltech-256* were not good, I decided to center myself on *Caltech-101*.

5. **Jump to *Keras***. In class, we saw a *Keras* example. As I could see, the code was much more readable and easy to understand, so I started to recode the program using *Keras* and *Python 3.6*.

6. **Error with shapes**. I had a lot of problems with the input shapes of my images (I was importing local files directly to the program). I spent here more than a week trying to figure out the error.

7. **Dataset change**. Tired of spending time on the shape error, I decided to use the *CIFAR100* dataset because the loading functions were already implemented.

4 Preprocessing of the images

To preprocess the images I use the *ImageDataGenerator* Keras utility. It randomly flips, applies filters, etc. to the images so that the input data is more varied. The parameters used are the following:

```
datagen = ImageDataGenerator(
    featurewise_center=False,
    samplewise_center=False,
    featurewise_std_normalization=False,
    samplewise_std_normalization=False,
    zca_whitening=False,
    rotation_range=0,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=False)
```

Figure 2: Preprocessing of the images.

5 Structure of the neural net

- Two convolutional layers with 128 filters using the ELU activation function.
- A pooling layer to reduce the size of the images.
- Two convolutional layers with 256 filters using the ELU activation function.
- A dropout layer to prevent from overfitting.
- Two convolutional layers with 512 filters using the ELU activation function.
- Another dropout layer.
- A flatten layer to convert the result to 1D.
- A fully connected layer.
- Another dropout.
- The output layer with 100 neurons, one for each class. The used activation function is softmax.

The optimizer that the model uses is *RMSProp*, with a learning rate of 0.0001.

6 The current state of the model

The last thing I did was to train the model with 200 epochs so that I could find the global maximum of the test accuracy. It took more than 8 hours on my local PC, but the results were quite good:

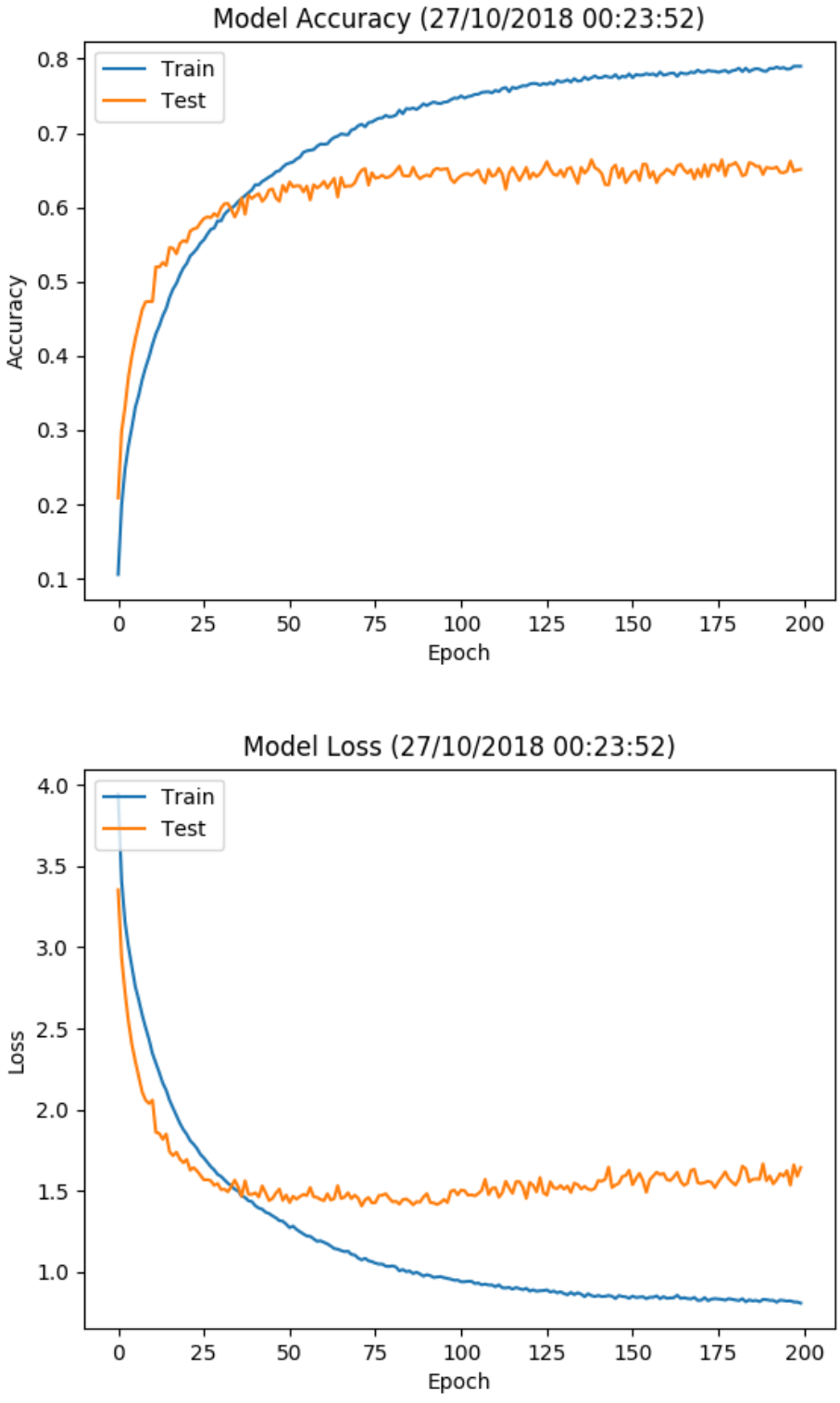


Figure 3: Accuracy and loss of the model

We can see that the accuracy is around 60%. Concretely, the maximum accuracy was 66.4%, reached on epoch 139.

7 The future of the project

The first thing that has to be done is to retrain the model with 139 epochs, so the maximum accuracy is reached.

During the next weeks my plan is to continue calibrating the parameters of the net (such as the learning rate, preprocessing, etc.) to try to achieve better results.

After that, I will test the net with images directly obtained from the internet. Another interesting option would be images which contains two objects, so that we can see if the net identifies one, both or none.

8 Examples

Let's run the model with different images retrieved from the Internet so that we can see its behaviour.

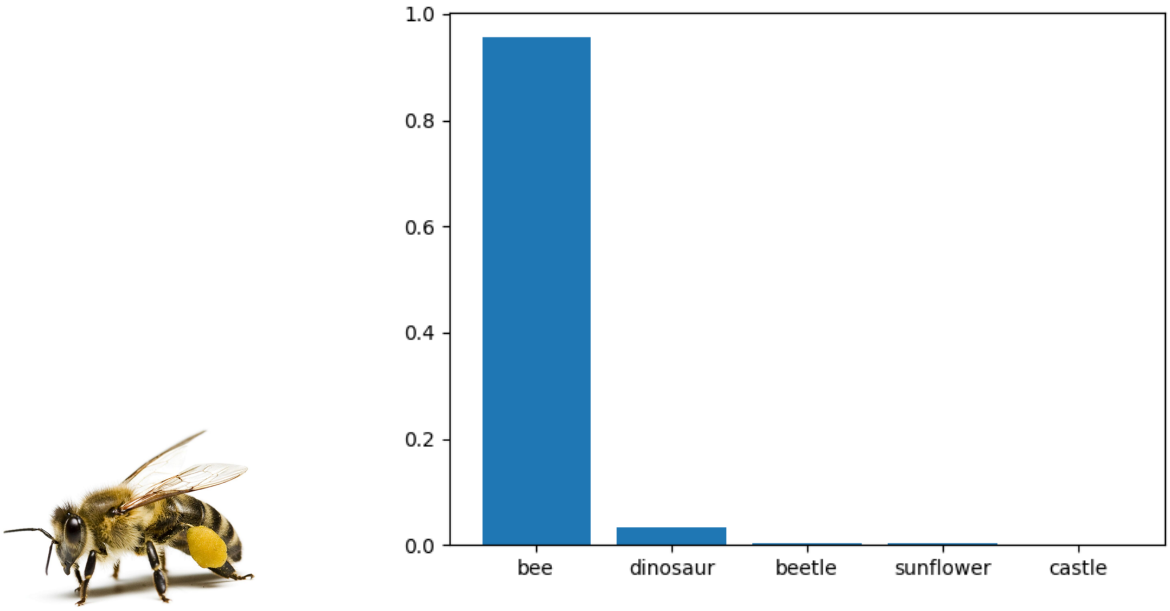


Figure 4: Bee prediction

We can see that the agent is succesful here. To check if the preprocessing has worked, we can flip the image and give it to the net.

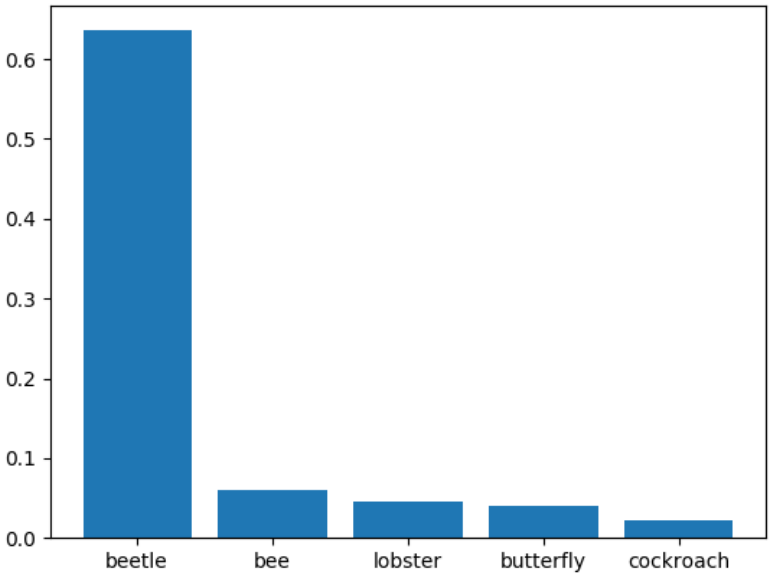


Figure 5: Flipped bee prediction

Although it is not the first option, the net recognised the bee. However, the preprocessing should be more intense.

Let's try other images:

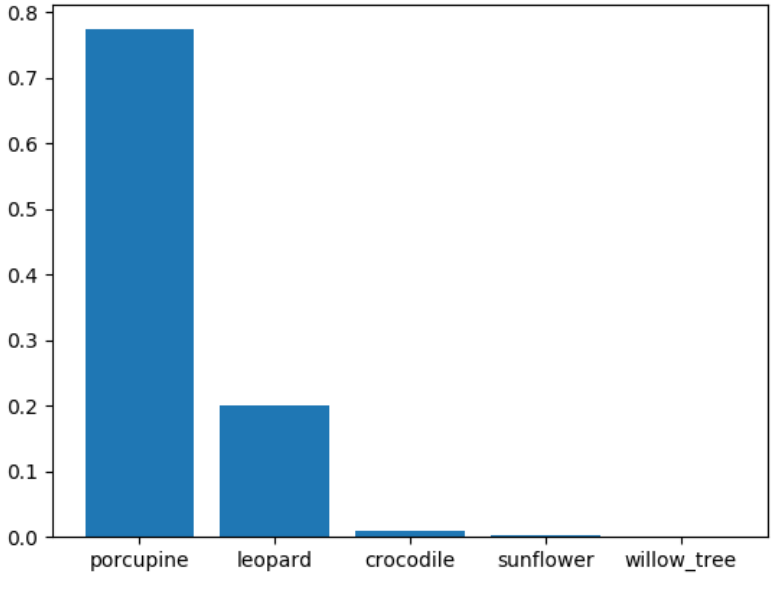
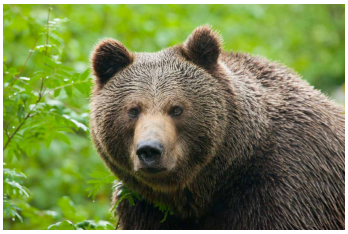


Figure 6: Porcupine prediction



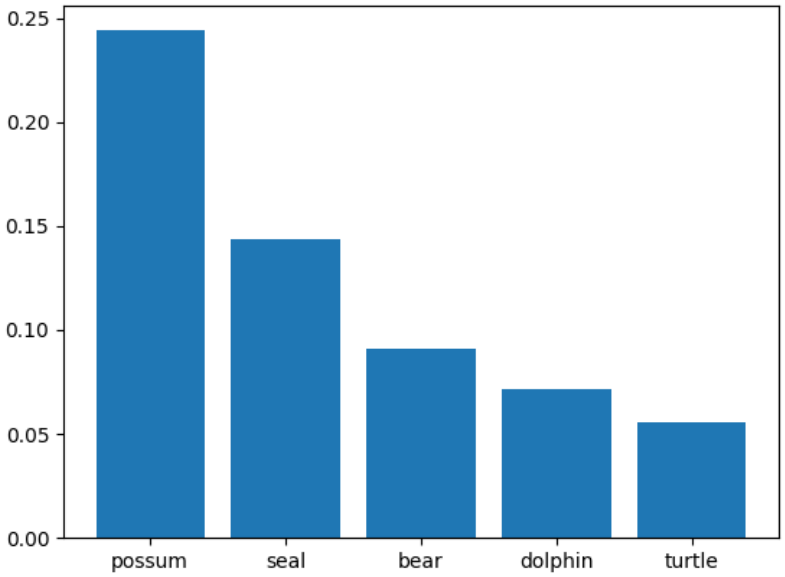


Figure 7: Bear prediction

9 Annex: classes contained in the dataset

1. apple
2. aquarium_fish
3. baby
4. bear
5. beaver
6. bed
7. bee
8. beetle
9. bicycle
10. bottle
11. bowl
12. boy
13. bridge
14. bus
15. butterfly
16. camel
17. can
18. castle
19. caterpillar
20. cattle
21. chair
22. chimpanzee
23. clock
24. cloud
25. cockroach
26. couch
27. crab
28. crocodile
29. cup
30. dinosaur
31. dolphin
32. elephant
33. flatfish
34. forest
35. fox
36. girl
37. hamster
38. house
39. kangaroo
40. keyboard
41. lamp
42. lawn_mower
43. leopard
44. lion
45. lizard

46. lobster
47. man
48. maple_tree
49. motorcycle
50. mountain
51. mouse
52. mushroom
53. oak_tree
54. orange
55. orchid
56. otter
57. palm_tree
58. pear
59. pickup_truck
60. pine_tree
61. plain
62. plate
63. poppy
64. porcupine
65. possum
66. rabbit
67. raccoon
68. ray
69. road
70. rocket
71. rose
72. sea
73. seal
74. shark
75. shrew
76. skunk
77. skyscraper
78. snail
79. snake
80. spider
81. squirrel
82. streetcar
83. sunflower
84. sweet_pepper
85. table
86. tank
87. telephone
88. television
89. tiger
90. tractor
91. train
92. trout
93. tulip
94. turtle
95. wardrobe
96. whale
97. willow_tree
98. wolf
99. woman
100. worm