



MEDIA INFORMATICS 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
Bibliography	9

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 bigger 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:

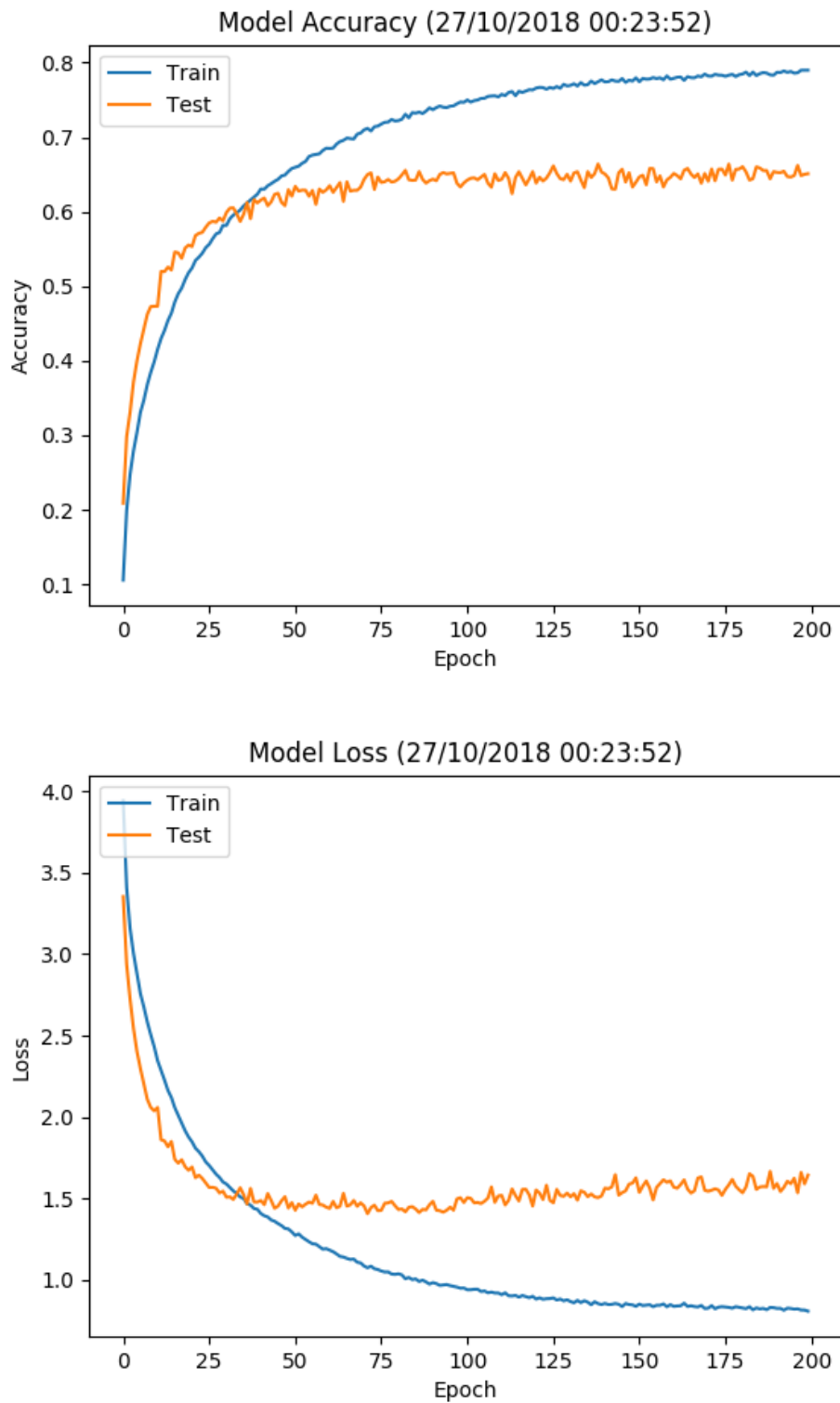


Figure 3: Accuracy and loss of the model

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.

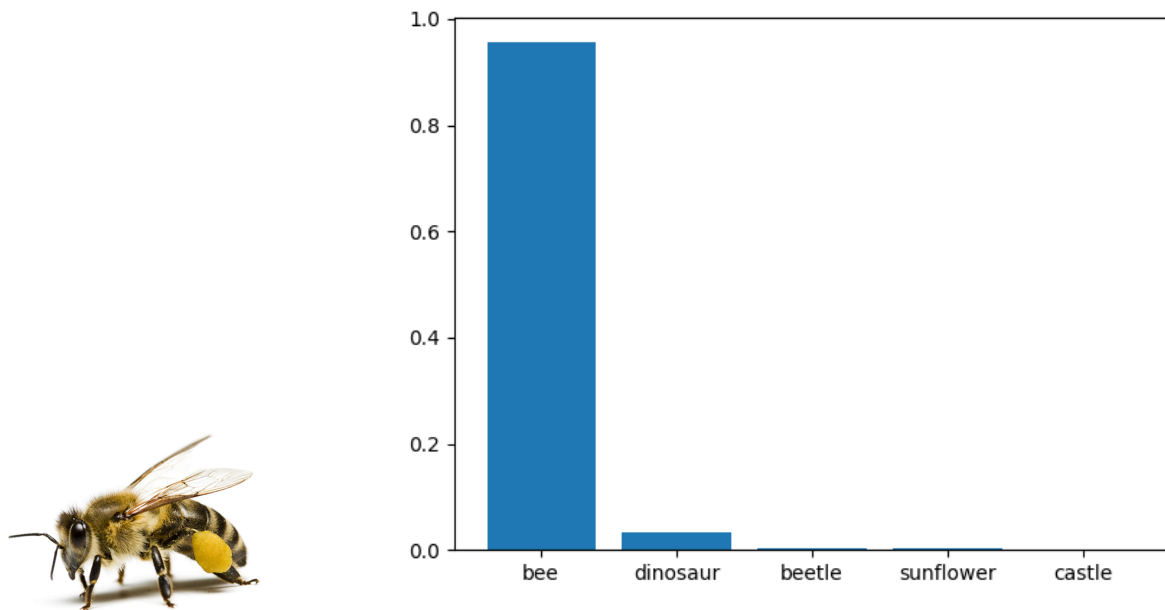


Figure 4: Bee prediction

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.

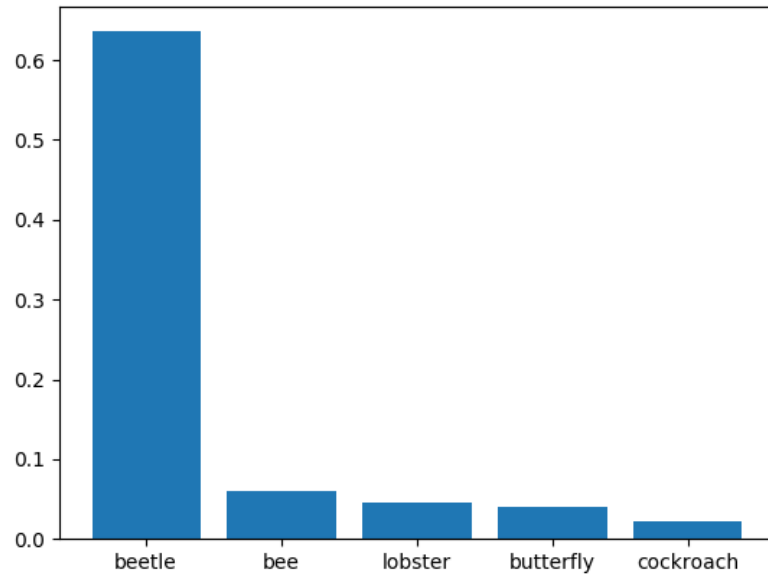


Figure 5: Flipped bee prediction

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

Let's try other images:

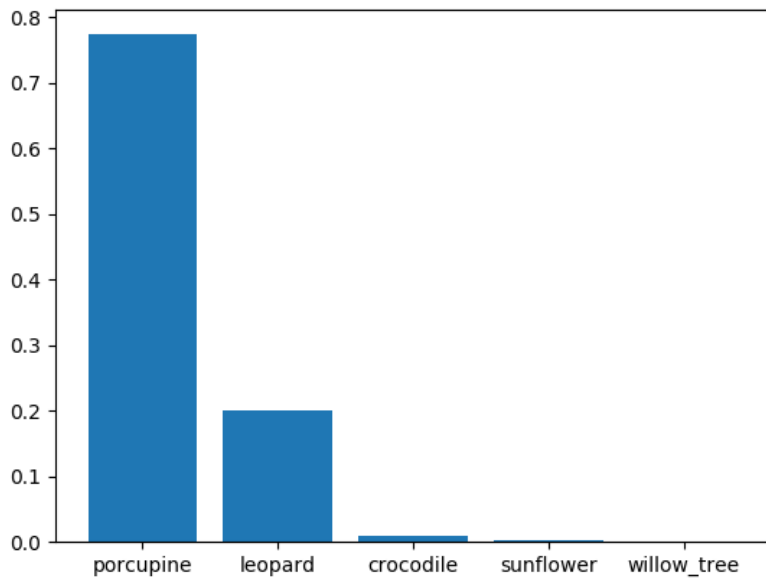


Figure 6: Porcupine prediction

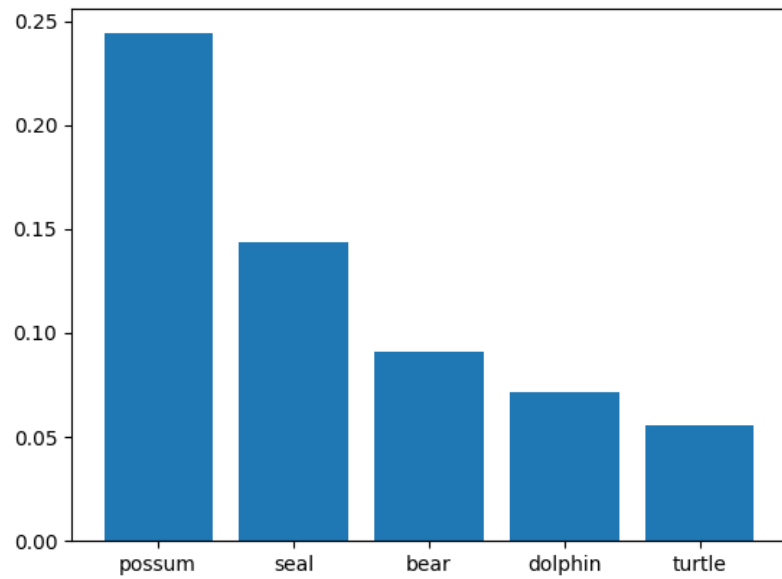
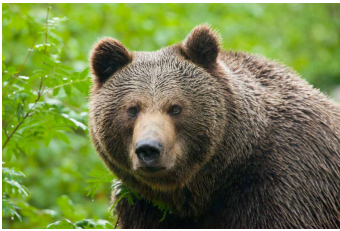


Figure 7: Bear prediction

9 Annex: classes contained in the dataset

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

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

Bibliography

- [1] Keras documentation
<https://keras.io/>
- [2] Moodle
<https://elearning.tmit.bme.hu/login/index.php>
- [3] Tensorflow Tutorial
<https://cv-tricks.com/tensorflow-tutorial/training-convolutional-neural-network-for-image-classification/>
- [4] Some Datasets
<https://www.analyticsvidhya.com/blog/2018/03/comprehensive-collection-deep-learning-datasets/>
- [5] Understanding Convolutions
<https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1>
- [6] Deep neural net tutorial
<https://medium.com/@tifa2up/image-classification-using-deep-neural-networks-a-beginner-friendly-approach-using-tensorflow-94b0a090ccd4>
- [7] Caltech-101 dataset
http://www.vision.caltech.edu/Image_Datasets/Caltech101/
- [8] Caltech-256 dataset
http://www.vision.caltech.edu/Image_Datasets/Caltech256/
- [9] TensorFlow tutorial
<https://cv-tricks.com/artificial-intelligence/deep-learning/deep-learning-frameworks/tensorflow/tensorflow-tutorial/>
- [10] Basic classification tutorial with Tensorflow
https://www.tensorflow.org/tutorials/keras/basic_classification
- [11] Keras tutorial
<https://medium.com/@vijayabhaskar96/tutorial-image-classification-with-keras-flow-from-directory-and-generators-95f75ebe5720>
- [12] Another Keras tutorial
<https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/>
- [13] Doubts resolution
<https://stackoverflow.com/>