Cmpe 462
2019 Spring

Machine Learning Class Project

Serkan Özel

What is it about?

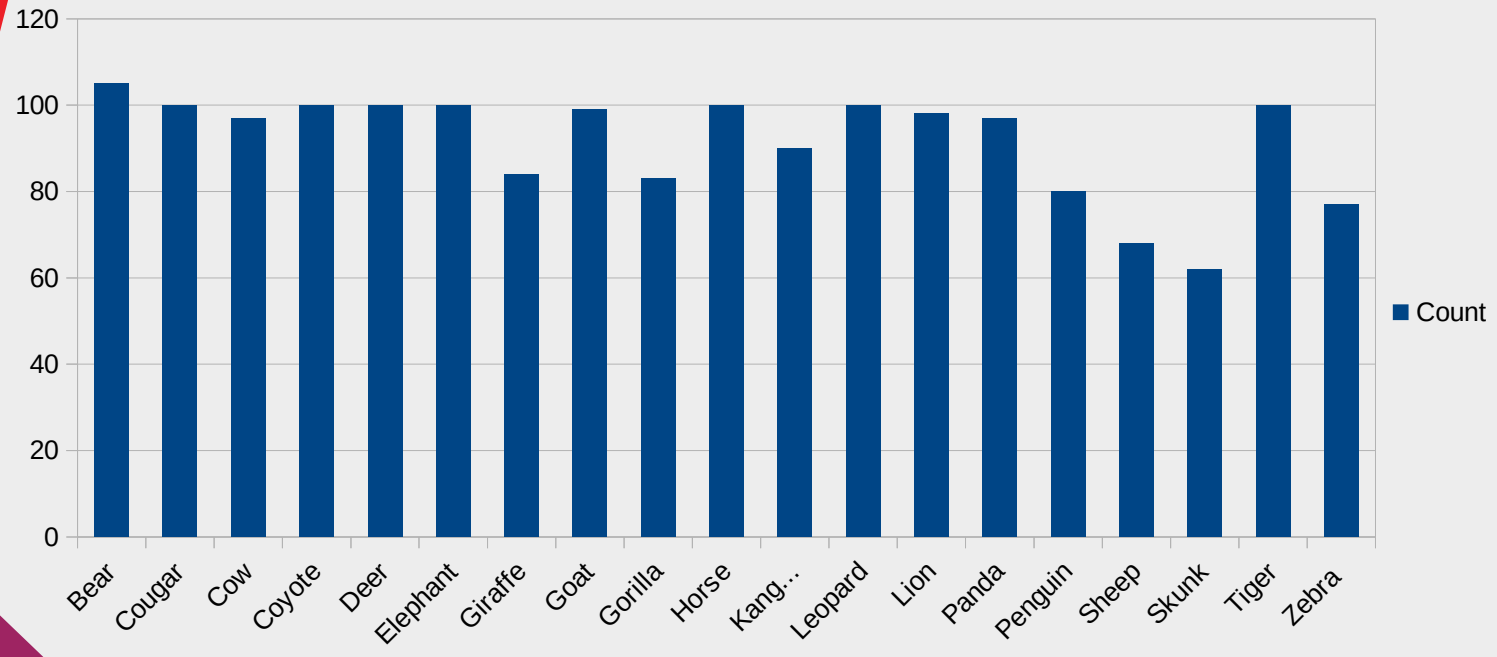
- I liked the idea behind the neural networks and wanted to implement it, experience and deal with problems that occur.
- Therefore, I used Keras to build an animal classifier with softmax with convolutional model.
- Building it from scratch was a good experience to learn what we can encounter when using a neural network.

What do I try to achieve?

- The dataset is taken from this site: http://www.csc.kth.se/~heydarma/Datasets.html
- It is called K-th animals
- It is a dataset from a KTH Royal University in Stockholm, Sweden.
- It is a challenging dataset with 19 different animals.
- I want to discriminate the animals in the test set with more than 80 percent success rate, though I get 50 percent only.

Some properties of our dataset

- Not all 19 animal classes have same number of images.
- All images can be of different size, typically near 250x250.



Some samples from my dataset

Bears

Cougars(Puma)

Pandas

Penguins

What did I do first ? How to handle the data?

- Images are preprocessed so that they are all of the same size (250,250).
- I split the the data set into two parts training, test set.
- I used 80, 20 ratio for splitting.

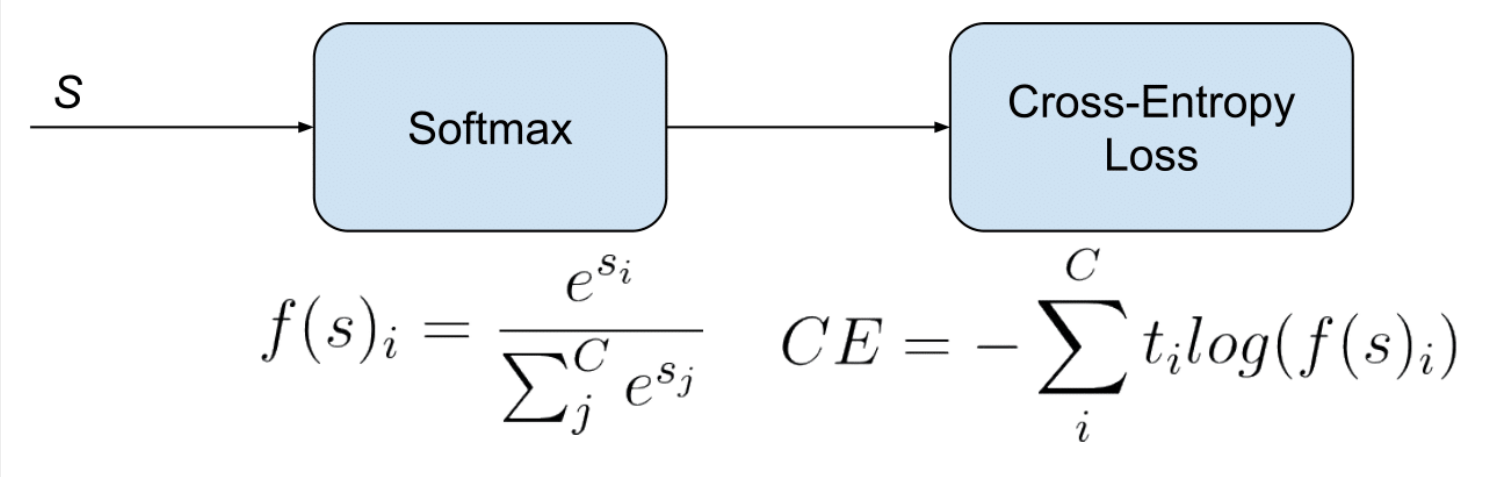
Softmax activation function

- Softmax activation function is useful for multi class classification.
- It produces a number between 0 and 1 for all classes and the numbers sum up to 1.
- These numbers can be thought as probabilities in terms of the class's percentage.

$$\text{Output for class } i = \frac{e^{i}}{\sum_{j=0}^{k} e^{j}} \quad \text{where i=0,1,....19}$$

Cross entropy loss function

- We used cross entropy loss – also called softmax loss – for assessing our the output of the model.
- t_i is ground truth value of i_{th} output.
- $f(s)_i$ is softmax output of i_{th} output

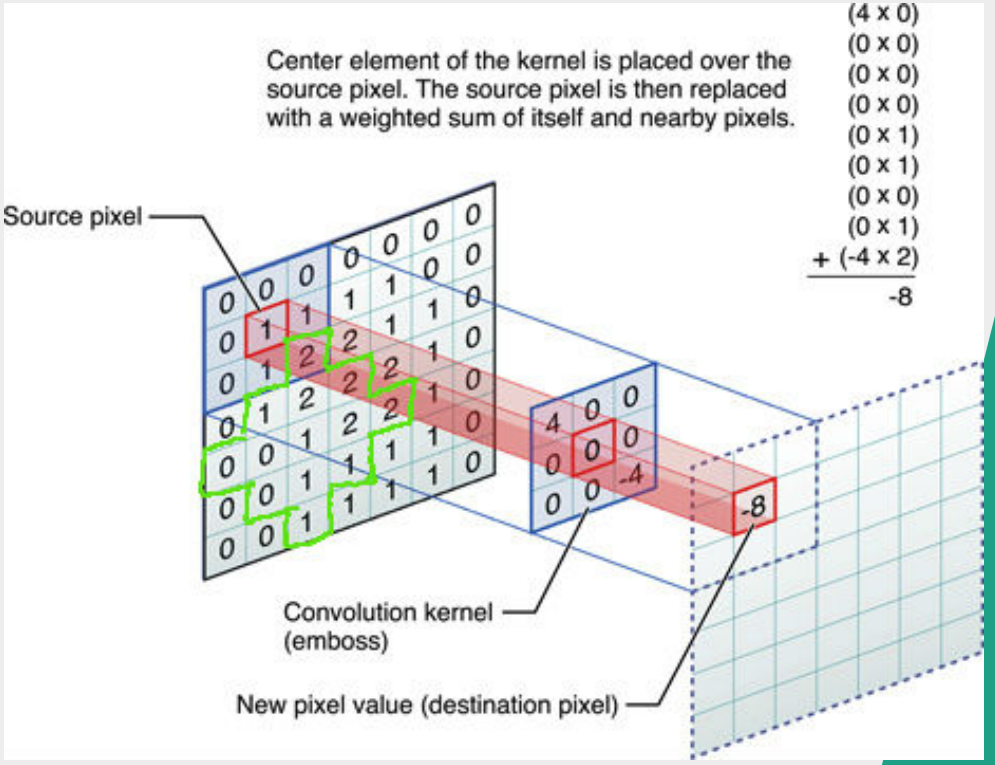


Stochastic gradient descent optimizer

- A gradient is basically the slope of a function.
- In SGD, it uses only a single sample, i.e., a batch size of one, to perform each iteration. The sample is randomly shuffled and selected for performing the iteration.
- SGD has one parameter in Keras: the learning rate.
- Its default value is 0.01.

Trials...

- First I tried to learn the animal classes by convolutional neural network, a CNN for short.
- When 3 layers of CNN used, 40 percent of test set accuracy is obtained with the Learning Rate = 0.01
- Convolution:



Result of 1st trial

- Though my training set accuracy increased up to 99 percent, this is not something good as it is called **overfitting**.
- In ML applications collecting more data is always useful, and I decided to use data augmentation to increase my test set accuracy.
- Data augmentation is modifying training data slightly, e.g pose, zooming in and out.. so that we have more data.

2nd trial: Dropout

- I add dropout layers, which is basically ignoring some units in the hidden layers randomly to reduce overfitting. i.e adding some randomness to evaluation.
- However, since it did not improved test set accuracy I give up using it.

Data augmentation

- I have used horizontal flip only. i.e 180 degree rotating the image horizontally.
- This made the space of input images bigger.
- After some trials, I realised data augmentation did not helped, so in the last configuration I have not used it.

Learning Rate Reduction Technique

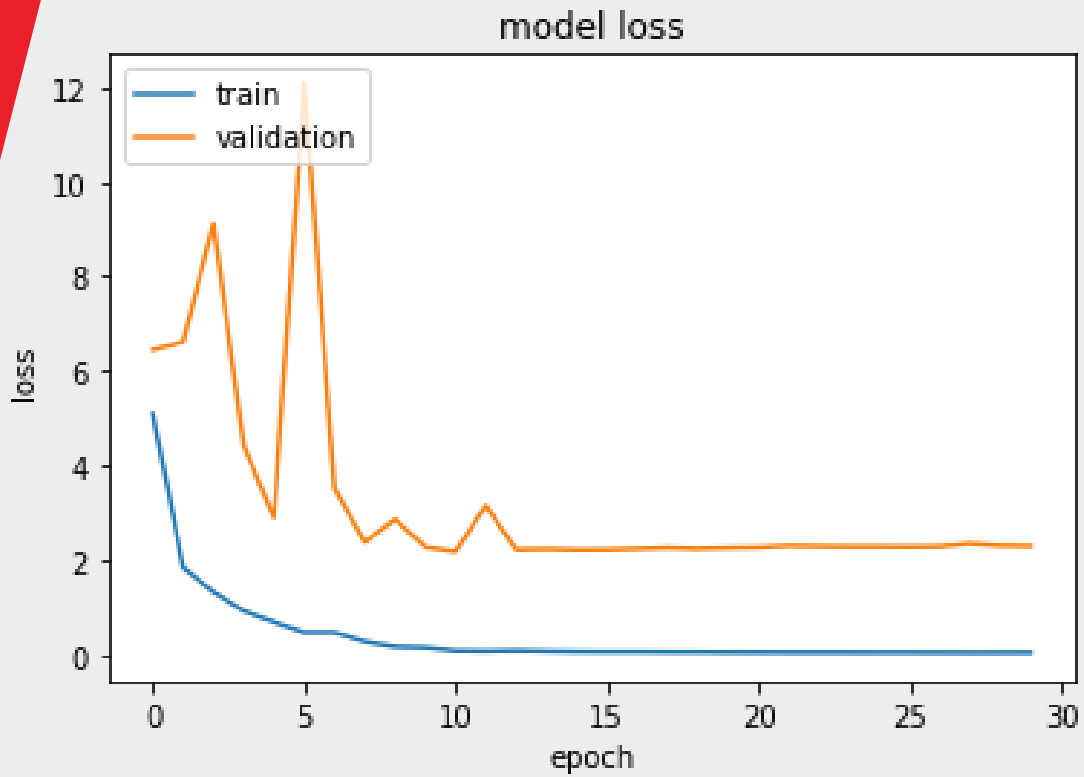
- As the algorithm become closer to the optimum point for the weights, it is important to lower the learning rate.
- Because we don’t want to go away from the optimum point, we take little steps there.
- I used this formula for learning rate where index is epoch number and // is integer division.

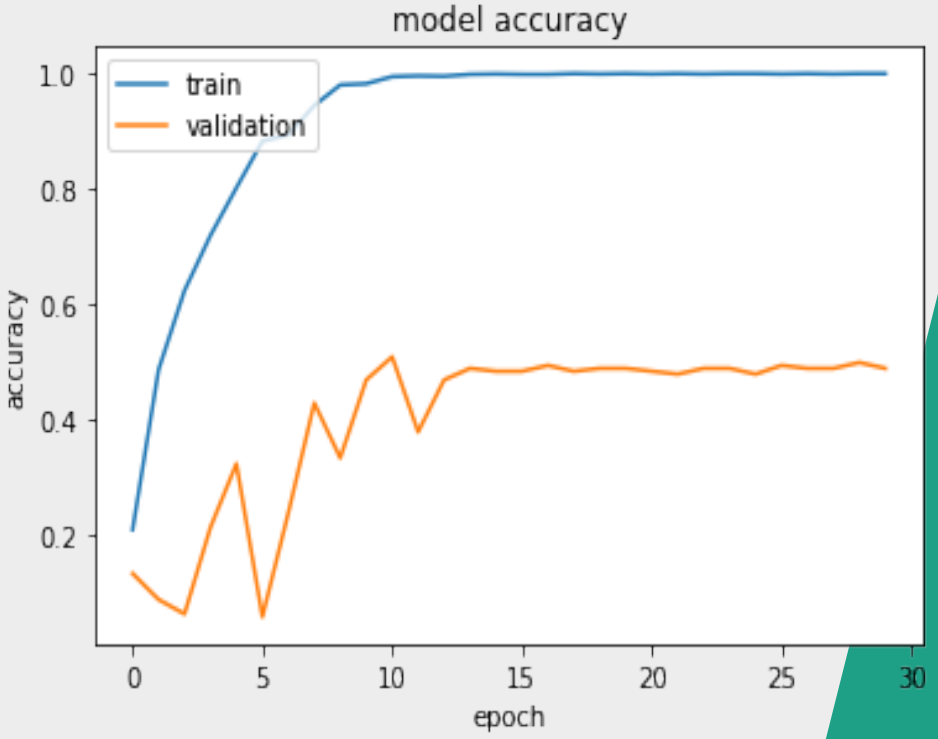
```
if (5*(index//10)) == 0:
    return 0.01
else:
    return 0.01/(5*(index//10))  # 0.002 when 10-20
                                 # 0.0005 when 20-30
```

Colab thoughts

- Because my pc does not have a GPU, I used colab to access a computer with a gpu.
- Using gpu increases the performance greatly.
- A drawback of colab is being have to upload the dataset to google drive or using git repositories to get the data to colab servers.
- I used the git method.

Some plots of the results





Results

- I realized that the success rate of test set is the best when the train set is overfitted.
- My algorithm’s success rate was:
- 100% on training dataset(we overfit the training set)
- 50.5% on test dataset
- after 30 epochs, with decreasing learning rate and without data augmentation.
- I think this is an enough success rate, because we don’t have a lot of data, and the data is not in super quality.

Thanks for listening

Serkan Özel