# Report

*Doggos emotions recognition and classification*

**Artificial Inteligence**
*Anna Przybycien, Agnieszka Szlendak*
May 24, 2019

**Abstract**

## Data preparation

In order to create a model for classification we needed proper dataset with dogs images. Unfortunatelly, for now, there is no proper dataset published suited for dogs emotion recognition. We were faced with the challenge of creating the dataset by ourselfes. The task consisted of four major parts:

1. Finding the data source.
   As downloading images manually from google graphics would be too much of a hussle, we decided to use a tool 4K Stogram, which enabled us to download pictures based on tags from Instagram directly on our discs.

2. Defining number and characteristics of classes.
   Because of the time and resource access constraint, we had to limit the number of classes to four. We also vaguely assumed what are the characteristics of each class:
   - **happy dogs** - dogs with the smile, mainly with open muzzle and tongue out
   - **angry dogs** - dogs that looks scary, with their teeth showing
   - **sleepy dogs** - dogs that are taking a nap, or are about to, with closed or squinted eyes,
   - **good dogs** - dogs that looks polite, with neutral muzzle

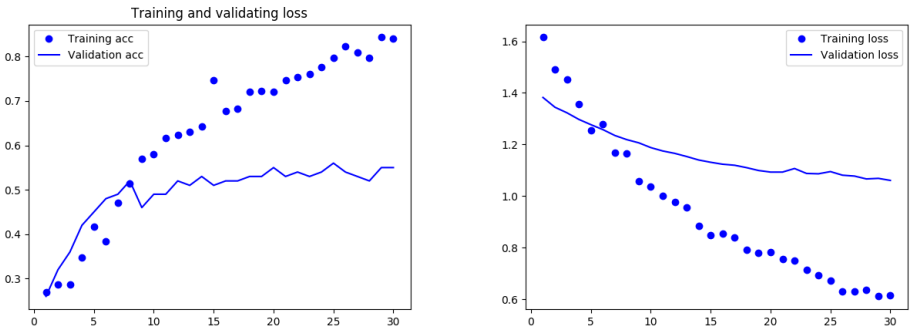
3. Determining classes numerousity & dataset creation.
   First, we assume that minimum number of images for each class should be 500. We began dataset preparation with the phrase 'the bigger the

better' in mind, but we were able only to reach our assumed baseline of 500 per class. So, from that point forward, we had dataset of 2000 classified images.

4. Seperating into train, validation and test sets.
   Lastly, as our dataset was small, we proceed to separate it using 60/20/20 proportion:
   - 60% - 1200 (300 per class) images in training set
   - 20% - 400 (100 per class) images in validation set
   - 20% - 400 (100 per class) images in test set

## AI implementation

- pure CNN
  We started with the simplest neural network we could think of, one consisting of four layers - every one used relu as its activation function - and one with softmax at the end.
  We went for 30 epochs and achieved 30% accuracy on validation data. Despite tremendous overitting we were off to a good start.
- CNN with data augmentation
  Overfitting was due to small size of our dataset, but it turned out that keras has perfect solution for it - easy to implement data augumentation - process that from one sample creates more by subjecting ir to various transformations.
  This time we decided on 100 epochs, so after some time (even using GPU) we got 56% accuracy with overfitting at smaller scale.
- CNN with convolutional base taken from 'imagenet'
  To counter overfitting resulting from using small dataset, we decided to use transfer learning and introduce convolutional base, pre-trained on famous imagenet dataset. Imagenet includes pictures of all sorts of animals, including various breeds of dogs, so we cocluded it would be suitable for our needs. It performed much better than pure version, with 60% accuracy and similar to the version with data augmentation.




(a) Accuracy function (b) Loss function

Figure 1: Accuracy and loss functions for CNN with 'imagenet'

- CNN with convolutional base taken from 'imagenet' with data augmentation and early stopping

## Final remarks

## Sources


- *Deep Learining with Python*, Fracouis Chollet.