CAPSTONE PROPOSAL

Domain Background:

Convolutional Neural Networks (CNN) are widely used in the field of AI and machine learning. Specifically they get fully into the field of deep learning.

Within this area it is common to solve problems related to computer vision; that is, the treatment of images in some way. That is why among the projects offered by Udacity to complete this nanodegree, there is one related to the aforementioned CNNs.

My personal motivation to develop the project in this line of learning is none other than being able to train in this field, so that's why I have chosen this project.

Problem Statement:

The problem to be solved is clearly defined since it is one of the pre-established options that Udacity provides.

In this sense, the project consists, broadly speaking, in developing a neural network model that, from an image given by the user, is capable of identifying the dog breed that appears in the image.

It also needs to be able to identify if a human is detected in the image, and then it will provide an estimate of the dog breed that is most resembling.

Datasets and Inputs:

Both human and dogs images datasets will be downloaded from Udacity platform.

The human dataset consists of 13,233 images of human faces. Normally only one face appears in these images, but this is not always the case; more than one could appear.

It is a dataset that, although it is labeled, these labels will not be used beyond the knowledge that a human's face appears in the image.

Furthermore, the dog dataset consists of a total of 8,351 images of different dog breeds. These images are labeled indicating the dog breed. There are a total of 174 different breeds labeled in 118 groups (some groups, due to the great similarity, encompass more than one breed) The dog dataset is already provided stratified and separated for training (80%), validation (10%) and test (10%).

Solution Statement:

The solution will be implemented with some CNN from scratch configuration, and also through transfer learning (possibly from a VGG-16, but perhaps Inception-V3 or GoogleNet)

In the creation of CNN from scratch, I will start working on training and validation datasets. Transformations will be applied to increase the sample and thus ensure that the network extracts patterns as general as possible (with maximum independence of scaling, rotation, translation, partial view)

Then, when creating the network as such, I want my start point to be a network that has at least three convolution layers with its Relu activation and its corresponding subsequent pool layers (maxpool), to end with at least two fully connected layers, with intermediate dropout and ending with a softmax activation.

With an initial configuration of this type I will do tests to see how far I can go and I will make changes in both the hyperparameters and the number of layers depending on the results obtained to try to achieve optimal performance.

This initial configuration and further changes are based on the original VGG16 paper '*Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)*' and other papers such as '*Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)*'. Nevertheless, my design will always be a freely and very simplified version of the original models stated in those papers.

Benchmark Model:

As mentioned above, a comparison will be made between CNN from scratch created specifically for this project, and one (or more) made from transfer learning from known models (VGG16, InceptionV3, GoogleNet)

Evaluation Metrics:

The evaluation metrics will be the ones commonly used; that is to say, checking the test accuracy. Specifically, the objective to be achieved is that the CNN designed from scratch is capable of having a minimum accuracy of 10% in the recognition of dog breeds.

This level of accuracy is a reasonable requirement in that it far exceeds simple guessing (lower than 1% with these data), and is not overly retrictive. Keep in mind that the task of assigning breed to dogs from images is considered exceptionally challenging, given the great similarity between some breeds whose distinction is difficult even for humans.

In the case of a CNN made with transfer learning the level of accuracy required will be at least 60% on the test set

Project Design:

The workflow for approaching the solution will be the one stated in the notebook of the Udacity's lab:

- Import datasets: Import them and analize size and structure.
- Detect humans: Use of OpenCV with its pre-trained face detectors.
  - Images are first converted to grayscale.
  - Write a function to detect the presence of a human face in the image.
- Detect dogs: Use of some pre-trained models to detect dogs (VGG-16, Inception-V3 and GoogleNet)

    - Normalization of inputs.
    - Predictions of outputs based on labels stated in ImageNet-1000.
    - Write a function to detect the presence of a dog face in the image, and being able to identify the correspondig breed (class label), using the pre-trained models.
- Create a CNN to classify dog breeds from scratch.
    - Write three different dataloaders (train, validation and test). This dataloaders will apply transformations (size, scaling, rotation, translation, crop, flip...) to the images used for training and validation in order to increase variability and this way obtain better adjust to the patterns and prevent overfitting. Batches will be defined and samples will be randomly shuffled.
    - Define an initial configuration of a CNN, with all necessary considerations about the convolutional layers (number of kernels, size, pad, stride), type of pool layers (size, pad, stride), fully connected layers.
    - Put this configuration in code with the corresponding activation functions, dropout to prevent overfitting, weight initializations and step-by-step definition of both the model and its behavior in the forward function.
    - Coding a training function that will take into account, in addition to everything previously seen, a loss function and an optimizer appropriate to the problem treated. Test different initializations and make a comparison between the fit of the training data and the validation data to find an adequate value of epochs to train.
    - Test the model and see if it meets the requirements. Try adjusting the hyperparameters based on the results obtained. Try different settings and approaches.
- Create a CNN to classify dog breeds using transfer learning. In this case the steps will be approximately the same as in the previous case, with the fundamental difference that in this case only the final layers will be redefined; that is, the CNN used will be that obtained by transfer learning and the final layers will be adapted to meet the requirements of this particular problem.
- Write my algorithm. Write an algorithm that accepts a file path to an image and first determines whether the image contains a human, dog, or neither. Then, if a dog is detected in the image, return the predicted breed. If a human is detected in the image, return the resembling dog breed. If neither is detected in the image, provide output that indicates an error. All of it, making use of the code developed before.
- Test my algorithm with different inputs and show some examples.