MACHINE LEARNING NANODEGREE

CAPSTONE PROPOSAL

Armando Contreras

07/27/2019

Domain Background

Computer vision is a subject that is at the forefront of a lot of machine learning applications and algorithms. I have a very personal interest in it since I have had the opportunity to work on mobile applications that use computer vision concepts and training to build interesting interactive experiences.

The task at hand is to create an algorithm that could identify and classify with a high degree of confidence an image as one of a predefined list of fruit types. We will use a dataset retrieved from Kaggle. Some machine learning algorithms perform very well for performing image classification, so during the project implementation there will be different tests to achieve optimal results. We will be following some of the guidelines to develop a CNN network supplied by PyTorch documentation and apply the learning on how to create the model layers. More information on the training data and research paper used in this project can be found in this research paper.

Problem Statement

We will be building a machine learning algorithm that will receive an image as input and will classify the image as a particular fruit. The image data set has been retrieved from a Kaggle dataset and the challenge is to find the best algorithm to attempt to accurately do image classification.

Convolutional neural networks are a technique that has excellent results when tackling an image classification problem. We can also test how pretrained models perform while classifying images. As a result, we will identify which algorithm is the best suited and performs the best with our given constraints.

Datasets and Input

The project uses a zip file that contains 114 fruit type classifications. Each of those classifiers has different images of the same fruit to help train our model. Our model will be trained on each of those image classifications and label the results appropriately

Input data fields:

- Class (defined by directory name)
- Images (list of files in a directory)

Input data characteristics:

- All images are 100x100px
- Images have 3 channels for color
- Image class with highest number of images is 900
- Image class with lowest number of images is 300

The total images in the training set are 57276, and 20% will be used for validation. The test set has a total of 19548 images.

The raw dataset has a class imbalance as noted above. We will try to get a more balanced set of classes through data augmentation.

We will also check the performance of PyTorch pretrained image classification models and check if the results are acceptable for our given constraints (reference attached)

Solution Statement

For this particular challenge, I believe that Convolutional Neural Network (CNN), will have the best results when doing image classification. We will try different training approaches to identify which models perform best for our classifier.

We will create a custom CNN from scratch and look at our model's performance. At the same time, we will check our results when using transfer learning with existing pretrained models. We will use image augmentation technique to improve our models performance. The input images used to train, validate and test are all 100x100 pixels.

Benchmark model

We will be benchmarking our performance against VGG-16. We will compare our results with the same input and evaluate our model's performance.

Evaluation metrics

The performance of our model will be determined by the logarithmic loss of the predicted class label. Our input classes are not balanced and therefore this will be a better indicator of good performance. Some of the information about why this metric will be used can be found in this article.

Project design

To start with our algorithm design, we will explore our dataset by looking at the images that are provided as well as the number of classifications that we need to classify. We will evaluate how balanced the classes are and proceed to understand how to process the input data before training.

We will use PyTorch to apply transforms to the input images. One of the reasons is to ensure all images are the same width and height, so we will apply a scaling transform. We also want to take into account rotation invariance, so we will also apply a rotation transform to our training set. Pytorch will also allow us to peform random horizontal flip to change the orientation of the image and improve traning.

To start working with our datasets, we will load our images into a few existing PyTorch pretrained models, VGG16 and ResNet, and evaluate the performance. Not all of the class labels exist in the pretrained models, so we will chose a few of the class labels that exist in both data sets and evaluate the performance.

We will also implement a CNN from scratch to compare our results. We will apply 4 or 5 convolutional layers and 2 fully connected layers to our input and measure our training and validation loss to tune our model appropriately and determine if it is performing well and fine tune. We will also include cross entropy loss function and keep track of our model with the best validation loss and store that as our optimal model.

Through trial and error, as well as experimenting with training parameters, we will optimize our solution to improve our performance metric.

References

https://pytorch.org/docs/stable/torchvision/models.html

https://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html

http://www.image-net.org

https://pytorch.org/docs/stable/torchvision/datasets.html

https://www.kaggle.com/moltean/fruits

https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

https://www.researchgate.net/publication/321475443_Fruit_recognition_from_images_using_deep_learning

http://wiki.fast.ai/index.php/Log_Loss