Udacity Machine Learning Nanodegree Capstone Project Proposal

Xian Wee

Domain Background

I have chosen to work on one of Udacity's proposed projects: classifying dog breeds. The subject falls under the research domain of image recognition, which received a major boost in 2012 when a team of researchers from the University of Toronto, led by the famed neural network pioneer Geoffrey Hinton, used a convolutional neural network ("CNN") to win the 2012 ImageNet Large Scale Visual Recognition Challenge. Subsequent competitions improved upon those results which propelled deep learning into the spotlight. Image recognition was one of the early "killer apps" of neural networks. Another was in the area of speech recognition where Hinton's research team demonstrated in a seminal 2012 paper that deep learning models substantially outperformed previous methods.

This project is interesting to me because of my aspirations to train a machine learning agent to "perceive" the world, eventually building a model of the world for the agent. One way of starting out could be to have an agent identify objects in the world and link them to words, much like a young child learns to associate simple words to objects - starting with simple nouns like "apple", then perhaps to verbs like "run" and "jump." From these building blocks, the next step might be to enable the agent to identify more abstract concepts. A richly textured model of the world could enable an agent to interact with it more effectively, to be able to make more accurate predictions and eventually, maybe begin to approach the major milestone of a machine learning agent exhibiting some semblance of "understanding" and "common sense."

Problem Statement

This project addresses classification problems using supervised learning. The inputs will be images of dogs and human faces. The application will determine if the image contains a dog (will classify image as dog/not dog). If there is a dog in the image, a convolutional neural network will take the dog image as an input and the output will be a prediction of the breed of the dog. If there is not a dog in the image, the application will detect if there is a human face in the image. If there is, a CNN will output a prediction of which kind of dog breed the human face

most resembles. If there is not a human the application will display a message indicating that neither a dog nor human was found.

Solution Statement

OpenCV's Haar feature-based cascade classifiers will be used to identify if a human face is in the image. To identify a dog in an image, I will use VGG-16, a pre-trained convolutional neural network, trained on ImageNet. Transfer learning with a CNN will be used to identify the breed of the dog or the kind of dog breed a human face most closely resembles. The type of transfer learning employed may be in the form of fine-tuning, where all of the model's weights are tuned during the optimization process or feature extraction, in which only the weights of the final layer of the model are adjusted through training. In this project I'll be using transfer learning in the form of feature extraction.

In project_research.ipynb the "Data Exploration" section provides details about the data. Each dog and human image in its raw form may vary in it's pixel dimensions. The images opened from their file path using Python Image Library, will be transformed to 224 x 224 pixels, converted to tensors and normalized. After the image transformations, a sample dog or human image will have dimensions 1 x 3 x 224 x 224 (torch.Size([1, 3, 224, 224])), meaning the first tensor contains 3 tensors representing RGB values of 224 x 224 image. Each pixel will be an input into the initial layer of the neural net upon which the first convolutional layer will apply filters.

Benchmark Model:

The performance of the CNN used to identify dog breeds or the dog breed the human most likely resembles will be compared against a benchmark model of a standard 3-layer feed forward neural network. The results of the benchmark model are in progress.

Evaluation metrics:

To evaluate the performance of the model I may look at accuracy, precision and recall. For this project, accuracy, or the number of images the model is able to correctly predict, would be an important metric because we would like to know if the model is capable of consistently recognize different breeds - a task even humans find difficult. Precision could be important as well - I would like to minimize false positives, meaning if the model identifies an image as a certain breed, hopefully the image of a dog is of that breed. Recall, or minimizing false

negatives, might be interesting to look at in terms of the cases in which the model assigned very low probabilities to breeds that actually turned out to match the image. I would want to dig deeper to see why the model was off by a lot in those examples.

Project design:

* Download the data and preprocess data by creating dataloaders using various transforms

    + Use Torchvision to create apply transforms and to create image datasets and data loaders (code for reference in *load_data* function in project_research.ipynb, under “Step 3”)

* OpenCV’s Haar feature-based cascade classifiers to identify if there is a human face in the image

    + Load color image, convert to grayscale, find the face using *detectMultiScale*
    + Result will be an ndarray containing the x, y coordinates of bounding box and width/height ([x, y, w, h]) (code for reference in project_research.ipynb, “Step 1: Detect Humans”)

* VGG-16 to identify if there is a dog in the image

    + Define VGG-16 pre-trained model
    + Pre-process the image, feed the image into the model and get the index of the highest probability output, which will indicate the type of dog breed (code in project_research.ipynb, “Step 2: Detect Dogs”)

* Transfer learning to predict the type of breed of a dog

    * Use a pre-trained CNN - resnet18

    + ResNet or Residual Neural Network (winner of 2015 ImageNet competition) is a CNN which contains “skip connections” to simplify the network by using fewer layers for training and to avoid “vanishing gradients” problem (the gradients in the early layers of a neural network become extremely small and the network cannot learn how a small change in the parameter’s value will affect the output)

    * Replace the final layer of the model using a linear transformation (nn.Linear)

    * Optimize the weights of the final layer through gradient descent/backpropagation during the training phase

* Integrate the various pieces of the project:

    * If there is a dog in the image, use the CNN to predict the kind of dog

    * If there is a human, use the CNN to predict what kind of breed the human most likely resembles

    * If there is no human and no dog, output an error message

Sources:

http://www.image-net.org/challenges/LSVRC/

http://cs231n.github.io/transfer-learning/

https://medium.com/limitlessai/2012-a-breakthrough-year-for-deep-learning-2a31a6796e73

https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/HintonDengYuEtAl-SPM2012.pdf

http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

https://www.kaggle.com/pytorch/resnet18

https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33

https://www.quora.com/What-is-the-vanishing-gradient-problem