Project Proposal

Domain Background

Pose detection is an application of Computer Vision to problems of real-time body movement tracking and human pose estimation. Between the models proposed to solve these problems there are PoseNet, DeepPose, and OpenPose. These methods work on finding coordinates of joints and key points within a person's body to identify the pose or movement. Real-life application of these methods includes gaming, augmented reality, and workout applications (AI-based movement coach) [2, 5, 7, 8].

Problem Statement

The focus of this project is to build a model that can detect yoga poses. And, in addition to create an web application which allows the user to input an image of a yoga pose and get its classification as response.

Differently from computer vision pose detection algorithms, which detects the relative position of key points, here the neural network is trained to recognize the shape and pattern of raw images. Further improvement of the actual model could consider its linkage (coupling) to pose estimators algorithms like PoseNet as done in [4], for instance.

Solution Statement

The solution adopted was to train the VGG16 image classification model on raw images to identify the yoga poses, i.e. to recognize image shape and pattern.

The model is trained and deployed in AWS SageMaker.

A web application is built to allow the user to use the model to classify an image containing a yoga pose. The application should be hosted via AWS Lambda and AWS API Gateway.

Datasets and Inputs

The dataset is available at kaggle [9]. It contains a total of 5994 sample images covering a range of 107 classes of images each one corresponding to a yoga pose.

During the process of data ETL images will be checked for file extension, file format, and image mode. A method for image mode conversion may be considered to comply with Torchvision Image Classification Models input image expectation [6]. The aim is to use only images in RGB mode to finetune VGG16.

Data ETL steps include: downloading the original dataset from kaggle using the kaggle API, extracting the dataset, exploring the data, defining the methods for converting image mode to RGB mode, testing the conversion method in a random sample of images, visualizing the output of the conversion, converting the full dataset (convert all the images to RGB mode), and finally compressing the actual data as a zip file and uploading it to Kaggle for public access and further use.

Benchmark Model

To use as benchmark model one of the solutions presented in kaggle to classify the 107 classes of yoga pose. The benchmark model is available at:
https://www.kaggle.com/code/kandhalkhandeka/efficientnet-pytorch-107-classes

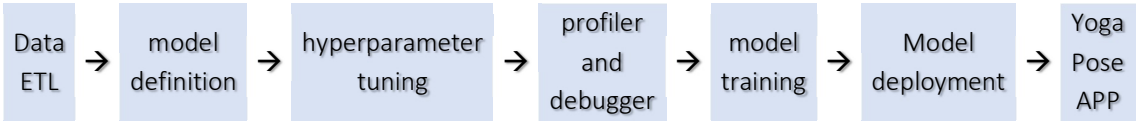
Evaluation Metrics

The VGG16 image classification model is optimized with Adadelta gradient descent optimization method [1] instead of the usual SGD optimization method.

The model performance is evaluated via test accuracy and average training and test loss.

Project Design

The project workflow should be as follow:



During data preparation (ETL) the dataset is downloaded and explored. Issues in file extension and file format are fixed and the images mode are converted to 3-channel RGB mode. As explained above, the dataset used to finetune the model in this project is composed of 3-channel RGB mode images only.

The dataset is then split into train, validation, and test sets (with the respective proportions of 0.7, 0.1, and 0.2). Manifest files must be created and uploaded to s3 together with the subsets of the data.

The model structure is defined as pretrained VGG16 with classifier layers of indexes 0, 3, and 6 unfreeze.

The model is then finetuned in AWS SageMaker using PyTorch as ML framework. Finetuning is performed in 4 steps for different ranges of the following hyperparameters: *epoch*, *batch size*, *test batch size*, and *learning rate*. The hyperparameters range should be defined as following:

```
hyperparameter_ranges = {
    "lr": ContinuousParameter(0.005, 0.015, scaling_type='Logarithmic'),
    "batch_size": CategoricalParameter([16, 20, 30, 40]),
    "test_batch_size": CategoricalParameter([10, 16, 20]),
    "epochs": CategoricalParameter([5, 10])
}
```

To enlarge the possibilities of hyperparameters being tuned a Warm Start Hyperparameter Tuning can be performed and further hyperparameter tuning jobs could be launched to narrow the range of values of specific hyperparameters depending on insights given by the results of the previous Hyperparameters Tuning and Warm Start Hyperparameter Tuning jobs.

After finding the best hyperparameters the model can be submitted to debugging using Amazon SageMaker Debugger API. This step is useful to identify possible training issues.

Finally, the model can be trained to be deployed.

Once the model is functioning, a web application can be constructed as an interface to the model endpoint. The web application will allow the user to use the model to classify an image containing a yoga pose.

The application should be hosted via AWS Lambda and AWS API Gateway with GET and POST methods configured to Lambda Function Integration type and Lambda Proxy Integration [3]. The Lambda function code is written in python 3.8 and will return a html content as response to a GET request and a JSON content, containing the model prediction, as response to a POST request.

Figure below shows a diagram of the application:

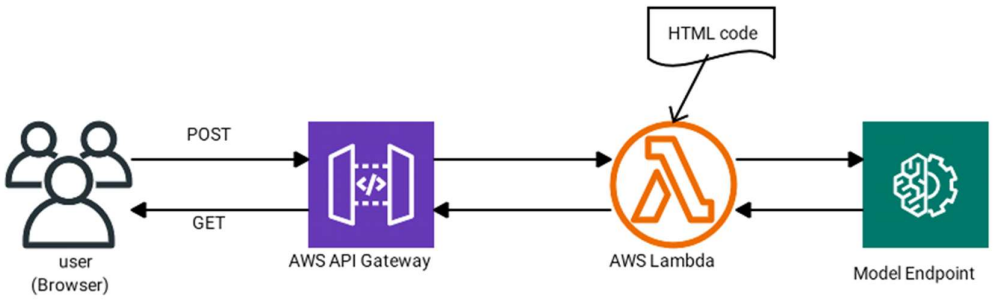


Figure 1: Yoga pose API diagram.

References

[1] An Adaptive Learning Rate Method. https://paperswithcode.com/paper/adadelta-an-adaptive-learning-rate-method

[2] A Comprehensive Guide to Human Pose Estimation. https://www.v7labs.com/blog/human-pose-estimation-guide

[3] Deploy Simple Web Applications in Lambda. https://aws-dojo.com/workshoplists/workshoplist47/

[4] ml5.js: Pose Classification with PoseNet and ml5.neuralNetwork(). https://www.youtube.com/watch?v=FYgYyq-xqAw

[5] Posture Detection using PoseNet with Real-time Deep Learning project. https://www.analyticsvidhya.com/blog/2021/09/posture-detection-using-posenet-with-real-time-deep-learning-project/

[6] PyTorch documentation. TORCHVISION.MODELS. https://pytorch.org/vision/0.11/models.html

[7] Yoga Pose Detection and Classification Using Deep Learning. https://www.researchgate.net/publication/346659912_Yoga_Pose_Detection_and_Classification_Using_Deep_Learning

[8] Yoga Pose Detection Using Deep Learning Techniques. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3842656

[9] yoga-pose-image-classification-dataset. https://www.kaggle.com/datasets/shrutisaxena/yoga-pose-image-classification-dataset