Human Activity Recognition Full Report

Menna El-Shaer

August 2020

1. Problem to be solved

In this project, we aim to solve the problem of recognizing human activities from 2D images. We do this by dividing the problem into two consecutive parts. Given new unseen activity images, first we estimate the location of body joints using a trained regression neural network, then we use the network's predicted estimates as inputs for a second neural network that has been previously trained to recognize the main activity in the image. We will evaluate the performance of both network models on a publicly available dataset. Deliverables for this project will be a text report detailing the methods and data analysis techniques used, the code and tools used to analyze, model and test the data. This will be in addition to a slide deck to be used for brief project presentations in the future. Solving the activity recognition problem is of interest to video surveillance applications while pose estimation is important for motion capture systems.

2. Dataset

We will use the MPII Human Pose Dataset, Version 1 available at: http://human-pose.mpi-inf.mpg.de/#download. The dataset consists of 24,984 images of about 40,000 people with annotated body joints. The dataset is already split (75%: 25%) into two independent training and testing sets. We will follow the same file split notations in this project. Each image in the dataset contains activities performed by one person or more. Activities are grouped into 20 categories with 410 classes or types. Each image is annotated with a single activity label that shows the main activity contained in the image.

In addition to the images and annotations, videos are provided for each image with one preceding and one following unannotated frame that can be used for additional model testing. Sizes of all images and annotations are 12.9 GBytes and 12.5 MBytes respectively.

3. Methods

A general overview of the project workflow is shown in figure 1. Following the traditional data science process, we start with acquiring and processing the data into an appropriate format for the main project tasks. We will use Python 3 packages to implement all project steps. The provided dataset annotations are stored in Matlab objects and structs which can be converted into Python

dictionaries easily using the scipy library. Since our main data are 2D images, convolutional network architectures are appropriate to implement our learning models.

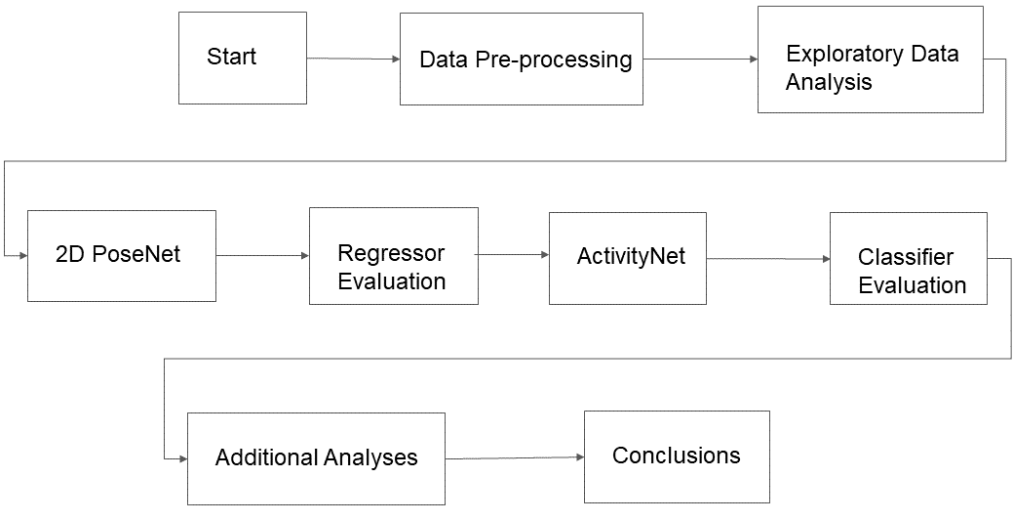


Figure 1: Project Steps and Milestones

3.1. Data Preparation and Pre-processing

3.1.1. Reading in the Data

There is a total of 24,987 images in the 'images' directory, which are split into training and testing by the image's corresponding binary "img_train" field in the annotations structure. Annotations are in a MATLAB structure format that are indexed by the image's location in the 'images' directory. Each image can contain one person or more that are individually annotated. Person(s) in an image are indexed by their own integer id in the image. Annotations are described as keypoints that represent the locations of sixteen joints of each person in the image. The annotated joints are a high-level representation of a detected person in the image; the Euclidean dimensions of the bounding box can be computed using the two-dimension positional coordinates of the joints. The sixteen represented joints are the left and right ankle, knee, hip, wrist, elbow, shoulder as well as the pelvis, thorax, upper neck and head top.

In addition, each image annotation has an activity field that labels the name of the activity in the image as well as the activity's category. Activity classes and category labels are formatted as strings. There is a total of 20 activity categories and 410 activity classes.

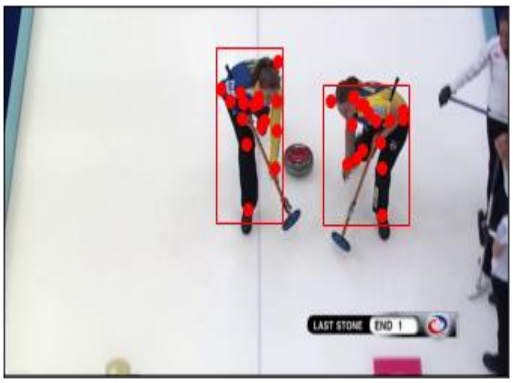
Before we begin the data exploration, it is necessary to get the data in a format that is compatible with Python libraries and deep learning frameworks. We used the “loadmat” function from the scipy.io library to convert Matlab structure fields into a Python dictionary keys and values. The resulting dictionary could be stored on disk as a .json format file for easier access. This process was done independently for the train and test files.

3.1.2. Data Cleaning

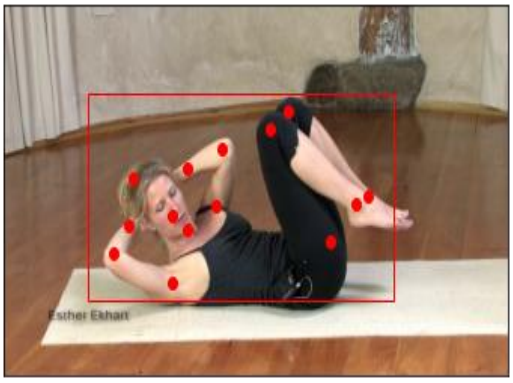
To check for missing person(s) annotation data for existing images, we defined a function that checks the length of the annotation list, if the list was empty for a detected person, we ignore that person’s annotation data. In addition, when extracting the bounding box dimensions for a detected person, we ignore invalid box dimensions i.e. negative widths or heights.

3.2. Exploratory Data Analysis

Out of the 24,987 images in the dataset, 17,503 are labeled as training images. Each image is labeled with one activity totaling 17,503 training labels or activities. Each image is also annotated with at least 16 keypoints -- ignoring missing data-- denoting the locations of a person’s joints indicating the body pose. This adds to a total of 401,470 training annotation points. Figure 2 shows examples of annotated keypoints overlaid on their corresponding images.


LAST STONE
END 1


101


Esther Ekhart

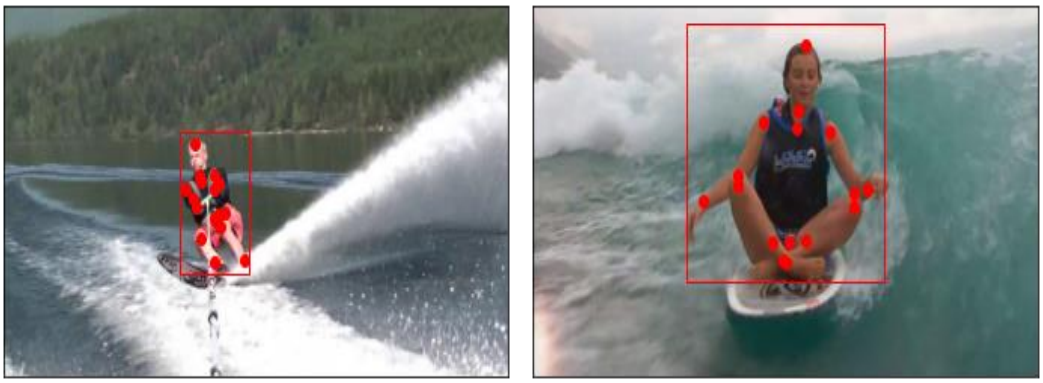


Figure 2: Annotated Training Examples

3.2.1. Problem Formulation

Our problem is divided into two parts: The first part is the regression problem of estimating the joint locations from an input image. In that problem, we treat the image pixels as independent variables with three color channels (Red, Green, Blue) and the pixels annotated as joints as the target variables. We have a total of 16 annotated joints resulting in 16 target variables or pixels; each pixel is represented spatially in the 2D x-y coordinate system and having a measurement of the color (RGB) intensity value.

The second problem is defined as a classification problem where the inputs are the 16 keypoints estimated from the previous step and the target variable is an encoding representing the activity class. It is interesting to note that there is a certain amount of correlation and independence between the input variables in this case and therefore, a Bayesian framework is appropriate if we were to model the input sample space as a Markov chain and use Bayesian inference techniques to determine the parameters of the distribution which in this case models the body pose or joints in an image.

We will discuss both problems and our solution approach in the following report.

3.2.2. Questions and Hypotheses

If we represent a detected person using the dimensions of its bounding box, we can formulate the hypothesis that the dimensions of the box (length and width) has no effect on the general activity category class. In other words, we would be asking whether there is a relationship between the general category of the activity e.g. sports versus music playing and the area of the bounding box computed using the body joint coordinates. The problem with using the size of the bounding box as a variable is that not all images are taken from the same depth distance, thus two people in two independent images could be performing

the same activity even though the horizontal and vertical dimensions (i.e. length and width) of the box are different due to foreshortening. Because in our image data, the depth is not a controlled variable, testing this hypothesis is futile.

We argue that identifying activity types in images using body poses alone is not enough, and that the context of the activity is important. Adding data variables about the environment to the detection model could help disambiguate the activity type, which is a good direction for future work.

An interesting question to explore is how the number of people in an image affects the activity classification. In other words, we can pose the following question: what is the correlation between the number of people in the image (assuming everyone in the image is participating in the activity) and the activity type? We can also state our null hypothesis as: The number of people in the image has no effect on the activity classification decision process. We will start by structuring our annotation data in a dictionary style format where the keys represent the activity class and the values are lists containing the number of people in the corresponding activity image map. The dictionary format is created for all labeled images in the training data subset comprehensively. We then compute the average (statistical mean) of the number of people detected in each activity class and plot the results. Figure 3 shows the results. We show the category class instead of the activity class since visualizing all 400 activities in one plot is not tractable.

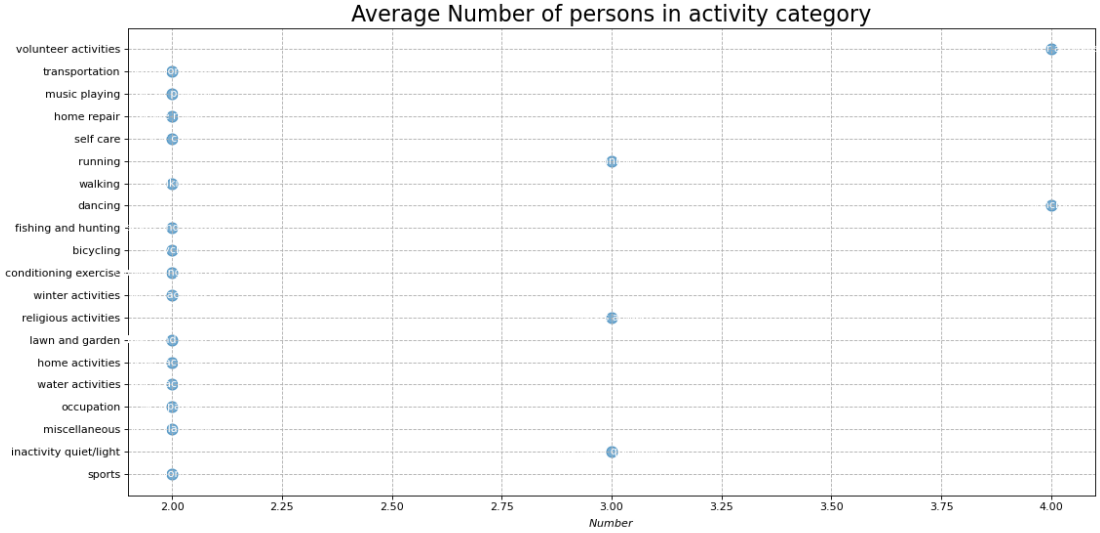


Figure 3: A dot plot showing the average number of persons participating in all 20 activity categories in the training dataset

To test the hypothesis, we can use the chi-squared test metric to test the correlation between the number of persons in a category and the category type. Since the activity category is an ordinal variable, the chi-squared test is appropriate to use to test the statistical significance of the correlation in question.

Running chi-squared hypothesis test resulted in a test statistic of 3.638 with a p-value of 0.99. Since the p-value is greater than the significance value of 0.05 which is the probability of obtaining results as least as extreme given the null hypothesis, we cannot reject the null hypothesis that the two variables are uncorrelated and our test result is statistically insignificant.

4. Deep Learning Modeling

In this section, we will discuss our machine learning models that we used the dataset for training. As mentioned previously, our model is composed of two sequential networks that solve the pose estimation and activity recognition problem. We will make use of fully connected convolutional networks with backpropagation to implement the process of learning from images.

4.1. Regression Network (2D PoseNet)

4.1.1. Data Pre-processing

After the data wrangling step, we have the training and testing annotations defined in a Python dictionary data structure that can easily be read from the saved .json files. We will format this dictionary structure into a Pandas dataframe that can be conveniently used in building our deep learning models. Before using the images for training, it was necessary to normalize and scale the pixel values as inputs to our model are expected to be in [0, 1] range. We also resized all images to 256 x 256 shape; as a result, the annotated coordinates for the keypoints also needed to be rescaled to the new image size.

4.1.2. Network Architecture

We will utilize a pre-built network model for visual representation learning that was trained on the ImageNet-1k dataset. The choice of using a pre-trained model on the ImageNet dataset was due to the comprehensiveness of the image database. It is generally used for object detection tasks since it contains more than one million images that were manually annotated with object bounded boxes representing the detected object's identity and location. Objects' types in the database span numerous categories and can be considered general enough for most object classification tasks.

We used the ResNet50-v2 network architecture as the feature extractor layer; it outputs a 2048-dimensional vector for each input sample (image). The model we used for transfer training was the BiT-S R50x1 [1].

We added a fully connected 32-output layer to the feature extractor layer to perform the keypoints regression task.

4.1.3. Training

Training with the full dataset was performed under limited time and computational resources, as a result, only one epoch of training was run. In addition, no fine-tuning of the feature extraction weights or any hyperparameter tuning of the network was done. We also did not perform any data augmentation techniques on the training data such as image rotations or translations or brightness changes. We aim to explore such options in future work.

The training process was done in batches of 32 samples (images) using the Adam optimizer and the mean-squared error loss function. The resulting accuracy after one epoch was 16.45%.

We used the trained model to predict keypoints to input into the classifier network which will be used for activity identification.

4.2. Classification Network (ActivityNet)

4.2.1. Data Pre-processing

As inputs to the classification network, we will use the x and y coordinates of the keypoints resulting in 32 input features. Activities in the dataset are grouped into 20 categories with a label for each activity/category. There is a total of 410 activities which will be mapped to 410 output labels. We will however use the activity categories instead of the activities themselves to fit our network. Using the individual activities for classification will be explored in future work. The categories of the activities are originally encoded as strings in the dictionary, which we will need to re-encode as binary labels for our multi-label classifier network. Each of the 20 category classes will be one-hot encoded into a unique vector for representation in the model.

4.2.2. Network Architecture

We will implement a straightforward multi-layer perceptron network with an input layer of 32 nodes fully connected to three 100-node hidden layers followed by a single 20-node output layer. We will use ReLU activations for the hidden layers and a softmax activation function for the output layer. To fit the network,

we use a categorical cross-entropy log loss function that is to be minimized by an Adam stochastic gradient descent optimizer.

4.2.3. Training Results

As with the regression network, we will train the model for one epoch (one pass through the training data) to minimize our time and computation resources. A single epoch training with the whole training dataset led to a training accuracy of 22%.

5. Future Work and Experimentation

The next logical step would involve some experimentation typical in deep learning modeling. First, the pre-trained weights we used for feature extraction and selection were used as is and not trained on our specific dataset. A next step would be to unfreeze the feature extraction layers and retrain using images from our dataset. We could also fine-tune and experiment with different hyperparameter values for the entire network when computing the best weights. We hypothesize this will improve our network's total prediction performance since the extracted features post training will be from the same dataset used to train the rest of the network. Second, a completely different architecture could be used for the feature selection process; we used the ResNet50 architecture in this project since it has been shown to work well on general images, but a different architecture could be better suited to images of human body poses specifically. Third, due to our constrained time and computational resources, we only trained the model for one epoch; a minimum of ten epochs or complete passes through the training data is usually done in deep learning modeling. Since training is done in an optimization manner, finding the best network parameters requires multiple passes through the dataset until convergence is reached. Consequently, multiple epoch training is necessary to ensure convergence. Finally, to prevent overfitting and ensure model generalization, evaluating the model on the test dataset and achieving a good prediction accuracy on the test data is crucial. We intend to conclude our project with model validation and performance evaluation.

References

[1] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly and N. Houlsby: Big Transfer (BiT): General Visual Representation Learning.