EVALUATION OF UTILITY OF 2D POSE ESTIMATION AS PREPROCESSING STEP FOR ACTION CLASSIFICATION

Viktor Zenk
AI Engineering Master Programme
Jönköping University
Jönköping, Sweden
zevi1720@student.ju.se

Willy Bach
AI Engineering Master Programme
Jönköping University
Jönköping, Sweden
bavu1616@student.ju.se

May 28, 2021

ABSTRACT

This study evaluates the utility of using an existing 2D pose estimation model as a preprocessing step instead of convolutional layers, as compered to a standard CNN model. Both approached use the Stanford 40 Actions dataset, containing images of humans performing 40 different actions. The CNN model is trained in a standard way, whereas for the other approach the images are first preprocessed to bring out the coordinates of 18 different body parts. These coordinates are then fed to a neural network which learns to infer from the now reduced and abstracted data.When the models are compared against each other, the pose estimation and neural network outperform the CNN in recognising human actions.

***K*eywords** human performance actions · classification · convolutional neural network · nerual network · deep learning · computer vision · artificial intelligence · pose estimation

1 Introduction

A Convolutional Neural Network (CNN) is a type of Deep Learning algorithm that is used in many applications, e.g. natural language processing, image classification, and image and video recognition. In image classification, CNNs are often the preferred Deep Learning algorithm because of its efficiency to extract spatial features from image input.

The convolutional layers in an image classifying CNN are designed to extract smaller features within the image. The model is left to find these patterns on its own, and hopefully finds something that can be used to classify the image. In some cases however, there are already proficient models and methods to extract useful features from the image, such as identifying body parts and their positions relative to each other.

This study evaluates two Deep Learning models' performance on action recognition; One standard CNN model, and a Neural Network (NN) which takes the 2D positions of 18 body parts, which have been extracted by a separate model. Both models are trained and evaluated on the same dataset, and their performances are compared.

1.1 Related Work

There are existing deep learning models that are able to detect the positions of human body parts in an image. One of them is presented in an article by Zhe Cao et. al. [1], which uses Part Affinity Field (PAF), a nonparametric representation as the approach, to associate body parts with individuals in the image. The architecture is designed to jointly learn part locations and their association via two branches of the same sequential prediction process. The method is able to identify 18 human body parts: the nose, neck, and the right and left side of the shoulders, elbows, wrists, hips, knees, ankles, eyes, and ears. It is evaluated on two benchmarks for multi-person pose estimation, the MPII human multi-person dataset and the COCO 2016 keypoints challenge dataset. The method presented by the authors won the COCO 2016 keypoint challenge and significantly exceeded the previous state-of-the-art result on the MPII multi-person benchmark.

The source code for the article has been made open source on GitHub, and this is the model which will be used to extract the body parts in this study.

The article by Amir N. et. al. [2] combined linear discriminant analysis with an Artificial Neural Network (ANN) for human action detection and recognition. Multidimensional features are generated using a linear discriminant analysis of the human body parts that are detected from the human silhouette. The generated features are used as inputs to an ANN which recognizes human actions. The method presented in the study is evaluated against three state-of-the-art methods where they all uses The Weizmann Human Action dataset and KTH-Dataset. The results of the study was that the proposed method got an accuracy of 87.57% on the KTH-dataset and 86% on the Weizmann Human Action Dataset. An accuracy improvement of 4.07% and 10.4% over the previously best results in KTH-dataset and Weizmann Human Action Dataset, respectively.

1.2 Stanford 40 Actions Dataset

The Stanford 40 Actions Dataset [3] is used to train both models presented in this study. It contains a total of 9532 images of humans performing 40 actions, such as applauding, running, or waving hands, and there are roughly 180-300 images per action class. The creators of the dataset also provide annotations in XML format for each image in the dataset. The relative information extracted from the XML-files are filename, action, and the bounding boxes of humans.

2 Methods

2.1 Convolutional Neural Network model

2.1.1 Preprocessing

The raw data from the dataset needed to be preprocessed before it could be used with Tensorflow and Keras to create the deep learning model. This is done by cropping the images using bounding boxes, resizing them to a standard 90x90 pixels, and normalizing the pixel values.

2.1.2 Convolutional Neural Network

The Convolutional Neural Network model used in this study consists of the following layers:

- Input layer - Convolution, 32 filters, 3x3, input shape = (90,90,3).
- Hidden layer - MaxPooling, 2x2.
- Hidden layer - Dropout, 0.2.
- Hidden layer - Convolution, 32 filters, 3x3.
- Hidden layer - MaxPooling, 2x2.
- Hidden layer - Dropout, 0.2.
- Hidden layer - Convolution, 128 filters, 3x3.
- Hidden layer - MaxPooling, 2x2.
- Hidden layer - Dropout, 0.2.
- Hidden layer - Convolution, 128 filters, 3x3.
- Hidden layer - MaxPooling, 2x2.
- Hidden layer - Dropout, 0.2.
- Hidden layer - Flatten
- Hidden layer - Dense, 256
- Hidden layer - Dense, 256
- Output layer - Dense, 40.

All layers in the model, except the output layer, use the same activation function (ReLU) and weight initialization scheme (He). The output layer uses the activation function SoftMax. Adam, a Stochastic Gradient Descent (SGD), was the chosen optimizer together with cross-entropy as the loss function. Accuracy is the monitored classification metric of the model.

2.1.3 Hyperparameter tuning

The hyperparameters were optimized using a rudimentary hyperparameter tuning experiment. The range of each hyperparameter to be tuned (number of convolutional layers, number of filters per convolutional layer, number of hidden layers, number of neurons per hidden layer, dropout level, and learning rate) was defined. From these ranges, 100 random combinations were selected, the CNN was trained on all training data using those settings, and evaluated to find its accuracy.

The best combination of hyperparameters found was saved, and was the one used for the study.

2.1.4 Compression

The model was also compressed into a tfLite model, in order to allow it to be run on a resource constrained edge device. This was done by first saving the model to a file, and then converting it to tfLite by using Tensorflow's built in tool to compress a saved model.

By using tfLite, the CNN model was compressed from 33.7MB to 11.1MB.

2.2 2D Pose Estimation + NN Model

2.2.1 Multi-Person 2D Pose Estimation

This study utilizes the open-source code [1] to generate 2D Pose Estimations coordinates of the Stanford 40 Actions Dataset [3] to be used as training and testing data for the NN model. A visual representation of the usage of the open-source code can be seen in Figure 1 below. Since the code can detect the 2D pose of multiple people in an image, a filter is applied to extract only the person with the most body part coordinates found.

These coordinates are then stored in a Numpy array, which can be used as input into a neural network. To accommodate the NN, the coordinates are normalized to values between 0 and 1, and null values (i.e. coordinates of body parts that were not found in the image) are set to 0.

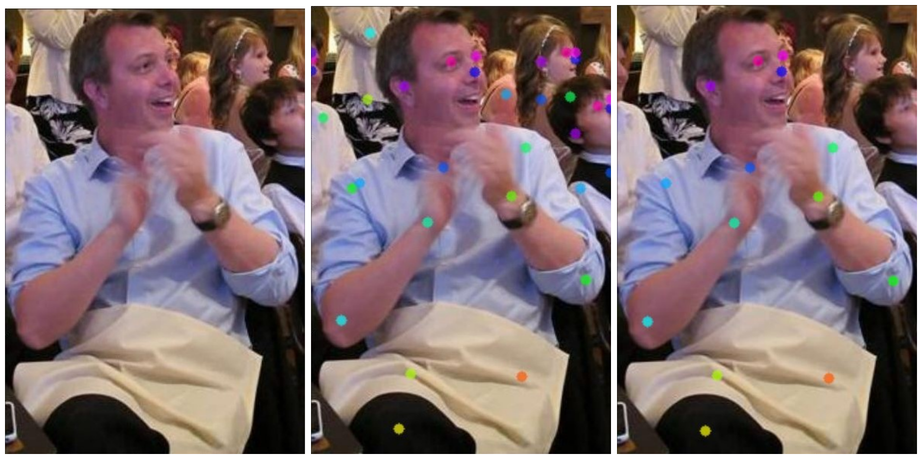
(a) Input (b) All found body parts (c) Filtered body parts

Figure 1: (a) Example input image (b) All body parts found, including some in the background which are irrelevant for this study (c) The body parts are filtered to only contain those of one person

2.2.2 Neural network

A neural network was created to take the 2D coordinates of the identified body parts as input, in the shape of a 2x18 Numpy array (x and y coordinates for each of the 18 body parts). From this, the NN attempts to classify the action associated with the image. The NN was made using the Keras API, and features four dense layers and a Softmax output layer.

2.2.3 Hyperparameter tuning

A similar hyperparameter tuning experiment to the one for the CNN was conducted to optimize the NN. For this model, the experiment was conducted in a slightly more efficient way, with all possible combinations of the hyperparameter ranges being generated. Since the number of combinations was too high for an exhaustive test to be tractable, a random selection of 100 combinations was tried.

In practice this is very similar to the approach taken for the CNN model, but with the advantage that the same combination is guaranteed to not be evaluated more than once.

2.2.4 Compression

The NN was compressed to a tfLite model using the same method as for the CNN. Due to time constraints however, the 2D pose estimation model was not compressed.

By using tfLite, the NN model was compressed from 1.19MB to 0.36MB.

2.3 Experiments

Each model was trained on 80% of the of the dataset, and the remaining 20% was used for evaluation. Both models, as well as their tfLite counterparts, were fed the evaluation data, and the number of correct and total predictions were gathered. From this, the models' accuracies were calculated.

3 Results

The results of the accuracy evaluations can be seen in Table 1 below.

	Baseline CNN	2D pose estimation + NN
Uncompressed	21.04%	31.52%
tfLite	18.36%	31.52%*

Table 1: Model evaluation results.

As is clear from these test results, the second method of incorporating the 2D pose estimation as a preprocessing step significantly outperforms the baseline CNN model, by more then 10 percentage points.

*Once again, it should be noted that only the NN was compressed, and not the 2D pose estimation.

4 Discussion and limitations

It can be noted that all the results are fairly low, compared to the accuracies often achieved in other image classification problems. The reason for this can only be speculated about, but one reason could be that the problem is quite a complex and abstract one. Compared to models that for example classify whether an image contains an instance of a class (e.g. 'person', 'car', etc.), the action classification models need to take it one step further, not only recognizing that the images contain people, but also *what those people are doing*.

Another reason that the models are struggling might be the high number of classes, without having the abundance of training data that might have been preferable for such problems.

The improved accuracy obtained from using 2D pose estimation instead of convolutional layers is an interesting result, and proves that such an approach can be useful. It needs to be stressed that this is no guarantee that this will always be the case as it will depend on many factors, not least regarding the problem at hand, and the accuracy and applicability of the feature extraction model used. For some problems, and perhaps this one as well, it is possible that a better CNN could be created that would outperform both methods presented in this study.

Due to time constraints, there are areas of the field of study which have not been possible to explore completely. For instance, it would have been interesting to see the effect data augmentation could have on the models' performance, and perhaps also reducing the problem scope to contain fewer action classes.

For completeness, it would also have been preferred to compress the existing 2D pose estimation model to a smaller format, so that it could also be run locally on a resource constrained edge device.

5 Summary and conclusions

In this study, the utility of using an existing 2D pose estimation model as a preprocessing step in classifying the activity of an image has been evaluated. A standard CNN was developed as a baseline, and trained on the Stanford 40 Actions Dataset. Due to the complexity of the problem, the accuracy achieved was modest, at 21%.

In comparison to this, another model was developed, which uses the 2D pose estimation model by Zhe Cao et. al. as a preprocessing step. This model was applied to the same dataset as used in the baseline CNN. The resulting dataset of body part coordinates was then used to train another neural network, allowing it to build inference patterns based on the now reduced and abstracted data. These two models together achieved an accuracy of 31%.

Using an existing model as a preprocessing step instead of the convolutional layers has thus in this case proved to be useful. However, the general conclusions to draw from this need to be carefully considered. In this case, the existing pose estimation model produced more useful abstracted data for the neural network than the convolutional layers did, but this cannot always be assumed to be the case. It is entirely possible that a better and more optimized CNN would outperform this result.

In conclusion, this method of extracting useful abstracted data using existing models can be viewed as another tool in the toolbox of an AI engineer, and while its utility in all situations has not been evaluated in this study, it has proved to be useful in at least one instance, and is certainly a method to consider.

References

[1] Zhe Cao and Tomas Simon and Shih-En Wei and Yaser Sheikh, "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields", *CVPR*, 2017

[2] Nadeem, Amir and Jalal, Ahmad and Kim, Kibum, "Human Actions Tracking and Recognition Based on Body Parts Detection via Artificial Neural Network", *2020 3rd International Conference on Advancements in Computational Sciences (ICACS)*, 2020, pp.1-6, doi=10.1109/ICACS47775.2020.9055951

[3] B. Yao, X. Jiang, A. Khosla, A.L. Lin, L.J. Guibas, and L. Fei-Fei. "Human Action Recognition by Learning Bases of Action Attributes and Parts", *Internation Conference on Computer Vision (ICCV)*, Barcelona, Spain. November 6-13, 2011.