Yoga pose recognition from still images and videos

Sobol J.
xsobol04
xsobol04@stud.fit.vutbr.cz

Hladyuk V.
xhlady01
xhlady01@stud.fit.vutbr.cz

Kabáč M.
xkabac01
xkabac01@stud.fit.vutbr.cz

Abstract

Convolutional neural networks are widely used in the domain of object recognition in images. These networks achieve good results for these tasks overall, but the development of new methods brings a new potential of upgrades that can improve accuracy even further. For our project, we have decided to use these convolutional neural networks in the domain of yoga pose recognition from still images as well as from videos. While there is a lot of people practicing yoga, obtaining a well made dataset ready to be used for training neural network models is fairly difficult. We have therefore decided to use a combination of multiple available datasets, as well as creating our own tool for creating new dataset to add for training and testing purposes. We have taken the well known Yoga82 dataset, and combined it with another dataset provided by prof. Herout. The model that we have implemented is a combination of two different models, one being based on DenseNet-201 pre-trained on ImageNet dataset, and the other being a joint position detection model. The combination of these two models enabled us to create our final model, which achieved decent results. We have also created an application, which allows us to speed up the process of new dataset creation from video recordings, saving our valuable time, and potentially helping other people in the future to create their new dataset to use for training of neural network models made to solve tasks similar to ours.

Keywords:Neural networks, deep learning, object recognition, yoga pose recognition, joint position detection

1. Introduction

Advancement in video processing and computer vision allows the usage of computers in many different sports. One of them is yoga, which is not used for research purposes as much as other, more popular sports. Development of convolutional neural networks allows an automation of the process of object detection and recognition, making it faster, more effective and more importantly it is usually fairly accurate. As a result, models using convolutional layers are a great option to use in the domain of image and video processing of sports and other activities, as well as recognition of human actions.

Since yoga is not one of the most common sports used for this type of research, there are not many datasets readily available for training of neural network models, which can impact accuracy of models, since not having enough training data affects the model's ability to generalize well, often times leading to model overfitting from training. Generally, convolutional neural networks take a simple image as an input, process it internally, and use it for either classification, object detection or recognition, or even other uses. Altering and changing training input can affect accuracy of trained models, making them more effective, and can have good impact on the overfitting problem.

The purpose of this project is to use convolutional neural networks for the task of yoga pose recognition from still images and from videos. Our main goal is to create a model that is capable of recognizing yoga poses from both still images and videos, with reasonable accuracy. Another objective is the creation of an updated dataset of yoga poses, which can be used to train other neural network models in the future. For this purpose, an application is created to take a video of a human practicing yoga, where our model takes

respective frames of these videos, and tries to recognize the yoga position depicted by the image with as high accuracy as possible.

Another goal of ours is to create an application that speeds up the process of creating new, similar datasets, which can save the time of people using it by being faster than manual labeling of data, with reasonable accuracy. For this purpose, the previously mentioned model will be used in combination with GUI application. The final product of this project will therefore be a new dataset created from video recordings by using our model and application, which can be used for similar tasks in the future.

2. Proposed solution

In this chapter we go over the data we use for training our model, their different properties, as well as the architecture of the model we implement to solve the task of yoga pose recognition.

2.1. Used datasets

Since yoga is not as common as many other sports used for similar tasks, we found out that it is fairly difficult to obtain good datasets to use for the training of our models. From the datasets that we managed to find, we saw that many of them are simply too small and not diverse enough, which usually leads to poor generalization of models due to overfitting problems. Therefore we decided to use a combination of multiple datasets to create one larger, more diverse dataset to use for training. For our main dataset, we have decided to use the *Yoga-82 dataset* created by Verma et al. (2020). This is a large scale dataset with a good pose diversity and good amount of different images of yoga poses, consisting of 82 different yoga poses. For these reasons it is one of the most commonly used datasets in the domain of yoga pose recognition, and therefore is a good backbone for our training and testing.

We have decided to combine the Yoga-82 dataset with another different dataset, which was provided to us by prof. Herout (herout@fit.vut.cz). This dataset consists of 11 different yoga positions, from multiple different people, taken in multiple different rooms with different backgrounds. What is interesting about this dataset is that for each pose sometimes multiple different images were taken at each time. One of these images is a basic side profile image. Other images of the pose are taken, but from different angles. Taking the side profile image as a base, these images use an offset from the base image, and are therefore taken from different angles relative to the base image, either from the left side or from the right side. This means that we effectively have multiple different angles of images of the same yoga pose, which should be much better for our training in order to improve the generalization ability of our model, as well as potentially improve the final accuracy of yoga pose classification.

Figure 1. A sample of images of a pose taken from different angles, taken from the dataset provided by prof. Herout

2.2. Data preprocessing

The datasets we decided to use were combined into a single bigger dataset. Since the smaller one of these datasets only has 11 classes, we have decided to only take the same 11 classes from the bigger dataset for training and testing of our model. However, in the future the remaining data could be also be used with a few small updates to the model, which would however need to be re-trained.

We have not used any special image preprocessing, but there is an option to increase the dataset size by augmentations in the future, which might provide better results. For the training and testing purposes, this bigger dataset was split into these standard parts:

- **Training data** - Consists of 70% of total images to train models.
- **Validation data** - Consists of 20% of the images for validation.
- **Testing data** - Consists of the remaining 10% of images. These were used to get the final accuracy results, as these images were not used in training process, meaning these images were unseen by model.

2.3. Architecture of our model

The simple, common solution is to use a standard convolutional neural network to take images as inputs, and get a class probability vector as output. This type of model usually can provide decent accuracy on its own. The other commonly used solution is using a program detecting join positions of humans, and using this as training inputs, which also usually leads to decent results.

Our proposed model architecture consists of a combination of these two types of approaches into a single, bigger model, as shown in figure 2.

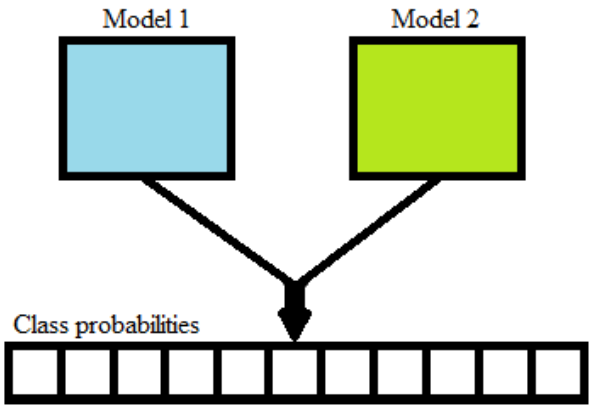

Figure 2. Basic illustration of our proposed architecture of our model

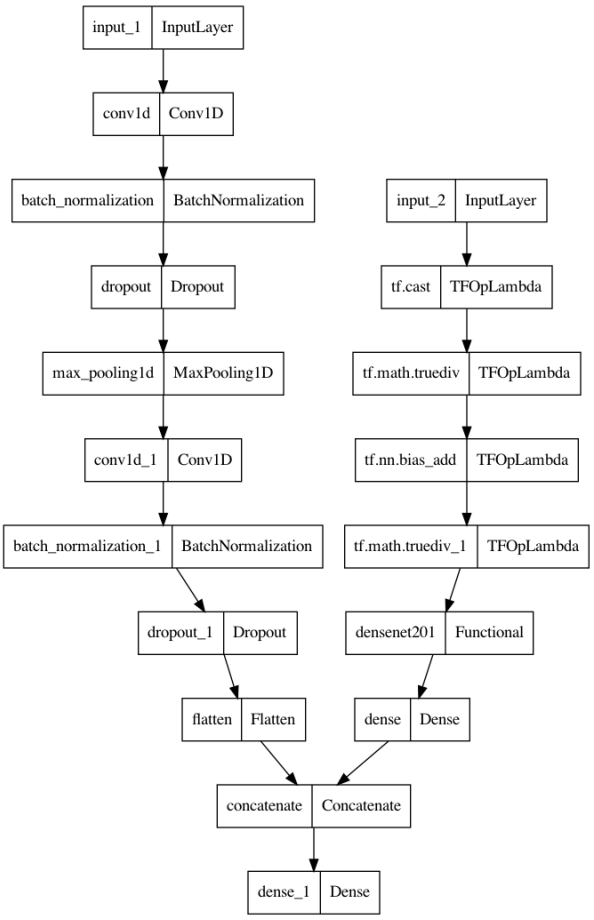

Figure 3. The final architecture of our implemented model

3. Implementation

Our model has been implemented in Python 3, with Tensorflow and Keras being used as a machine learning library for the creation and training of the model itself. The final architecture of the implemented model can be seen in figure 3.

The left branch represents the architecture of a joint detection model. Here, a vector of joint positions of the human on input image is created, which consists of 33 locations of the respective joints, represented as [x, y] coordinates. We have used the library for Python (2022) to extract these coordinates of joints. The output of the last dense layer here is a vector of 256 values.

The right branch represents a traditional type of the convolutional neural network model, where classic image is given as an input. This input is then passed through DenseNet-201 network made by Huang et al., 2016, which is a 201 layers deep convolutional neural network used for problems in similar domains. This model has been pre-trained on the ImageNet dataset created by Deng et al., 2009. Output of the last dense layer of this branch is a vector of 256 values, which is then fed forward towards the final output layer.

Before the final output layer, the vectors from both of these models are concatenated, creating a vector of 512 values. This vector is then finally fed into the last dense layer, which is the final output layer consisting of 11 values, each representing the probability of its respective yoga position. There the maximal value is found, its index is mapped to its yoga position, which is then used as a result.

4. Model testing and results

The implemented model has first been trained using images only from the *Yoga-82* dataset. On this data, our model managed to achieve an average testing image accuracy of 94% overall, calculated from 10 different runs. Then we have decided to combine our datasets to create the bigger dataset to be used for our model training. By using this new dataset as

well as hyper-parameter tuning of our model, we have achieved 95.5% overall accuracy over 10 runs, with the best model having an accuracy of 97% on our testing images. Therefore we have decided to include the saved models from this run as our solution. These models can therefore be also loaded to be used again.

5. Active learning and our application for active learning

Active learning is a method of machine learning where an algorithm can interactively query an annotator for to label input data to classes. Completely manual data labeling is a very time consuming process and requires people's time and energy. Active learning is an algorithm used for querying data to annotator (or teacher) to label them automatically. There are multiple variants of active learning. The goal of active learning is to create more data that can be added to a dataset, increasing its size and variability, which can in turn lead to improvements of neural network models in the future thank to training them using more labeled data than before. An illustration of the active learning is shown in figure 4

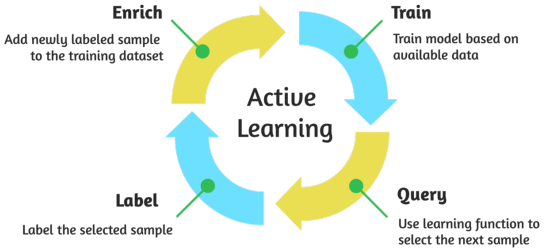

Figure 4. An illustration of the process of active learning in neural network models

In our project we have focused on efficiency and simplicity of use, therefore we have created an application for automated labeling of data.5 The input of our application is a video recording of a yoga session. The application does pre-processing of these yoga videos and queries for classification model with images to classify these inputs based on the highest confidence of each of the 11 available yoga positions. The application is simple to use. We can save images, move to subsequent images, or return to previous images, as well as an ability to delete images if they were saved incorrectly. Of course, if our model fails to classify the input image into one of the available yoga positions, the user can simply choose the correct position and save the input image as the right class, into the respective class folder. We aimed to make the application as simple to use as possible, just so that the application can be used on various types of data to label, not just yoga position images.

Figure 5. Screenshot of application

5.1. Experiments for the labeling application

To put this all into a perspective, we have done experiments with manual data labeling and labeling with our application. First variant consists of us sampling a video recording and labeling the data manually. The second variant consists of us using our application, which is used for the same video recording as the previous variant. Conditions of experiment:

- Video length: 2 minutes 30 seconds, 60 frames per second.
- Sample rate of manual labeling variant: 2 images per second.
- Sample rate of app labeling variant: 3 images per second but actual query time to teacher is lower because model filters more useful images.
- Pre-processing time of app labeling: this varies greatly based on the capabilities of the machine that is being used to run the application, therefore in the final results this time is excluded.

The time saved by using our application in comparison to manual labeling can be seen in table 1

The conclusion of the results of these tests is: We can see that labeling using our application is **3.8** times faster than manual labeling. Using manual labeling, we have labeled a lot of data, but this took a lot of time, since we need to label each frame of the video individually. Since a lot of the frames that are close together in the video are basically very similar with minimal changes,

Table 1. Comparison of data labeling between our application and manual labeling of data

	Manual labeling	App labeling
Samples	9 000	9 000
Analysed samples	300	450
Saved samples	140	117
Time of labeling	6:12	1:19
Sample per minute	23	89

a lot of time is wasted. Our application allows us to skip frames, which can save a lot of time and therefore outperform manual labeling, since a lot more video recordings can be processed in the same time period. Based on these results we think that our application can help and improve the process of dataset expansions thanks to the use of the active learning method.

6. Conclusion

The objective of this project was to classify images and video recordings of people performing yoga positions into their respective classes. We have managed to implement a model by merging two models together to create a new big one, that is able classify inputs into one of the 11 available yoga positions, with decent accuracy. We have also managed to train this model on more data thanks to merging two existing datasets together to improve the results, and on top of this we have also created a whole new dataset consisting of about 1000 new images of yoga positions, which can be expanded even further, and then used to either improve our model even further, or in new models requiring similar data for training. This new dataset was created from video recordings of yoga sessions, using the application that we have created, which is faster than manual labeling, and therefore saves valuable time. The application offers a wide range of applications other than yoga position classification, which can help other people in the future solving similar tasks, by giving them the ability to create new datasets for training their neural network models.

References

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255. https://doi.org/10.1109/CVPR.2009.5206848

Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2016). Densely connected convolutional networks. https://doi.org/10.48550/ARXIV.1608.06993

library for Python, M. (2022). *Mediapipe library for python*. Retrieved May 26, 2022, from https://google.github.io/mediapipe/

Verma, M., Kumawat, S., Nakashima, Y., & Raman, S. (2020). Yoga-82: A new dataset for fine-grained classification of human poses. *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 4472–4479.