Visual Computing Project Final Report
Yoga posture images classification using Histogram of Oriented Gradients

Mathilde Larchevêque
CentraleSupélec
mathilde.larcheveque@student-cs.fr

Carlos Santos García
CentraleSupélec
carlos.santos@student-cs.fr

Abstract

In this project, we show that Histogram of Oriented Gradient features combined with classical machine learning systems can compete and even outperform more complex deep learning-based methods on the task of Yoga posture classification, showing that classical computer vision techniques are far from being obsolete.

1. Introduction

Yoga is a group of physical, mental, and spiritual practices that originated in ancient India and aim to control and still the mind, recognizing a detached witness-consciousness untouched by the mind and mundane suffering. It has increasingly become one of the most popular disciplines throughout the world. Given an image of a person performing a Yoga posture, we would like to classify his/her pose. This feature could be useful for online Yoga classes, very common after the Covid-19 pandemic, and could improve personalized learning through fine-grained statistics about one's training. Such a system could give interesting insights to personal coaches about their pupils, and improve everyone's mental health.

To the best of our knowledge, no research has been conducted to solve this problem with efficient algorithms that could be embedded on low-resource hardware. We will rely on a classical computer vision algorithm, mainly HOG feature extractor, to try to solve this problem.

2. Problem definition

Given a set of labelled images $D = (x_i, y_i)_{i \in I}$ where $x_i \in \mathbb{R}^{H \times W}$ and $y_i \in \{0,1\}^K$ where K is the number of classes in our open-source dataset ($K = 47$), the objective of our project is to find a simple and fast classification technique using a mapping $f_\Phi : \mathbb{R}^{H \times W} \to \{0,1\}^K$ with trainable parameters Φ so that:

$$f_\Phi(x_i) \approx y_i$$

The mapping f_Φ will rely on combining the HOG feature extractor introduced in [4] with a classifier with trainable parameters.

The objective is here to assess whether simple extracted features can already provide enough information to compete with some more complex and computationally expensive models based on deep learning architectures such as Xception ([3]), EfficientNet ([8]) or BigTransfer ([5]) for which results are available here. Note however that these models have been trained by particulars on Kaggle, so the comparison will only be able to suggest how these different models can perform compared to classical computer vision techniques. We will focus on two particular metrics (see 5.3 for details) to make the final comparisons.

3. Related work

Feature extractors are commonly used in computer vision to reduce the dimensionality of an image, making it easier for a model to identify important features and patterns in the data. We here present some of the most common feature extractors in computer vision, and explain our choice for our particular problem of classifying human poses.

3.1. Feature extractors

Feature extractors are useful for classification tasks in computer vision because they allow the algorithm to focus on the most informative and relevant parts of an image, while disregarding less important information. Raw images are high-dimensional objects with noisy and redundant information that can prevent a classifier from learning a correct way to perform any particular task. Feature extractors have been widely used : the idea is to extract a reduced set of features that are more informative and less redundant than the raw image pixels. These features are used to represent the image and are fed to a classifier, which uses them to make predictions. Additionally, feature extractors can also be used to make images more robust to changes in viewpoint, scale, and lighting conditions. This allows the classifier

to be more robust and generalize well to new images.

There are several commonly used feature extractors in computer vision for classification tasks. Some of the most popular include:

- Histogram of Oriented Gradients (HOG, [4]): This algorithm captures the shape and structure of objects in an image by computing the gradient orientation of pixels and creating a histogram of the orientations.
- Scale-Invariant Feature Transform (SIFT, [7]): It detects and describes local features in an image, which are invariant to image scaling, rotation, and changes in viewpoint.
- Speeded Up Robust Features (SURF, [1]): This method comes as an improvement over SIFT, it is faster and more robust to changes in viewpoint and scale.

All of these feature extractors are useful in different scenarios, and the choice of which one to use depends on the specific task and the characteristics of the images being used.

HOG features seem however especially pertinent for human pose classification, since they capture the shape and structure of objects in an image. This task is core for distinguishing different yoga poses, which can involve subtle changes in the position and shape of body parts. They were first introduced to allow human detection in images, a closely related problem to ours. In addition, HOG features are not very sensitive to small changes in viewpoint, scale, or lighting conditions. According to the authors, it does not even require filtering before applying the algorithm. This makes it robust for our classification task. In comparison, features like SIFT and SURF are more suited for object detection in images, and can be more computationally intensive than HOG features. We will therefore mainly base our study on using HOG features to perform the classification.

3.2. Image Classifiers

Image classification is the task of assigning a label or class to an entire image. In the era of deep learning, the extraction of features was fully automatized by convolutional neural networks, as Krizhevsky *et al*. introduced in [6]. However, before this deep learning era, two of the main classification techniques used were:

- Feature-based methods: these methods relied on extracting informative features from images - such as colour histograms, texture features, and shape features - and then using these features to train a classifier. Due to the high-dimensional nature of images, extracting meaningful features from the images is usually an effective way of representing the image in a smaller but still informative way.
- Template-based methods [2]: these methods relied on comparing an input image to a set of predefined templates and identifying the closest match.

Overall, these methods were not able to achieve the same level of accuracy as deep learning methods, but can provide a much smaller inference time on low-performance hardware and usually require much less computational power for training.

Figure 1. Raw images of 12/47 classes the dataset

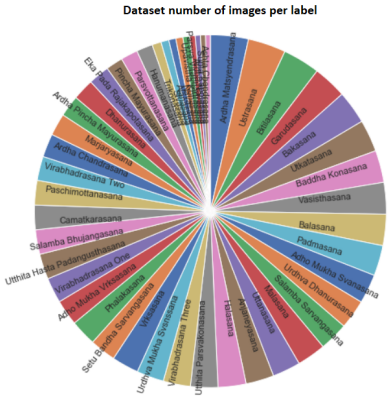


Figure 2. Repartition of images per label

4. Methodology

4.1. Dataset

The Kaggle Yoga Posture Dataset - used for training, validating and testing - is composed of 2758 images, divided into 47 classes. The number of images per classes is imbalanced, as some classes have five times more images than others. The repartition of the classes is shown in figures 2 and 3. The median size of images is 316x344.

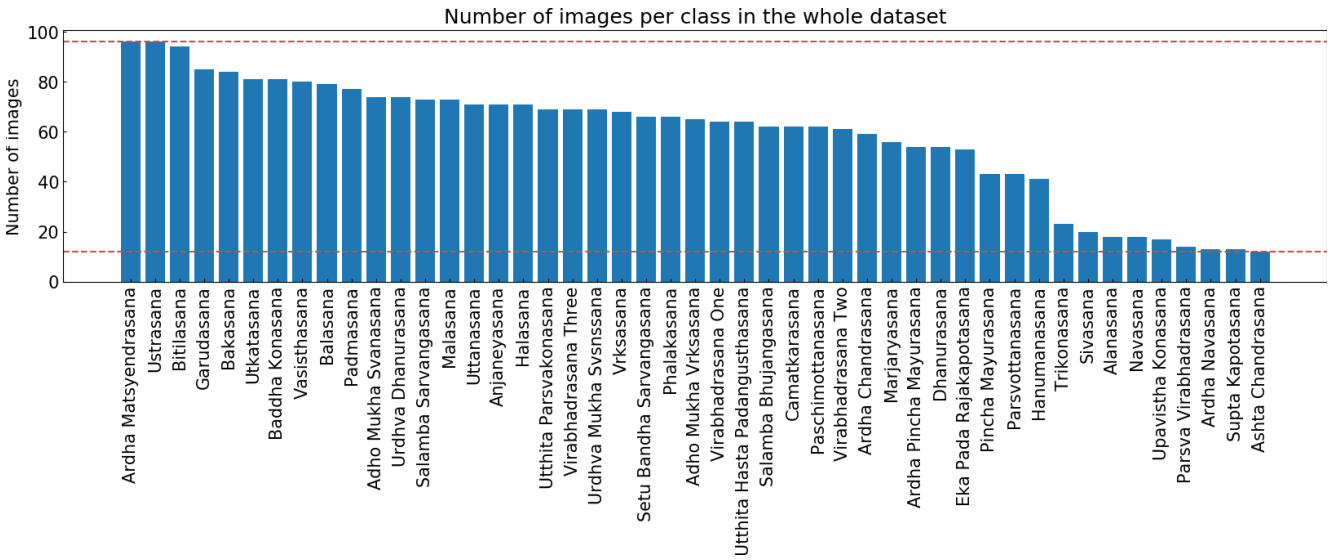


Figure 3. Number of images per label. The most frequent class contains 96 images, while the less frequent class only contains 12 images.

4.2. Data Augmentation

As seen in figure 3, some classes of the dataset contain less than 20 images. Considering the fact that there are more than 1000 features used for the classifier, we use data augmentation to improve the training dataset. Data augmentation techniques can be divided into two categories: position augmentation and colour augmentation. However, some constraints due to HOG properties and the dataset images must be taken into account: HOG is a colour invariant feature extractor, and important image rotations ($> 30°$) may change the yoga posture. As a result, the data augmentation techniques that were used for the training set were:

- horizontal flipping
- $\pm$ 20° rotation
- zoom
- positive and negative shift of (5, 10) pixels (out of (60, 80))

Classes with less than 30 images got their sizes multiplied by 7, while the classes with more than 30 images were only augmented using Horizontal flips (multiplying their size by a factor of 2). Figure 4 shows the new number of images per class in the training set.

4.3. Histogram of Oriented Gradients

Histogram of Oriented Gradients (HOG, [4]) algorithm is a technique in computer vision that works by computing the gradient orientation of pixels in an image and creating a histogram of their orientations. This histogram is then used as a feature descriptor for the image, which can be used in machine learning algorithms for object detection or image classification. The HOG algorithm is particularly useful for image classification tasks because it is able to capture the shape and structure of objects in an image, which is important for distinguishing between different classes in our case.

Our feature extraction pipeline is the following: first, we preprocess images by *border padding* them, *i.e*. replicating the borders of the image to match the target shape. The image is then resized to the size of 60×80, which we empirically found was enough to reduce both space and computational complexity of our pipeline while maintaining a good level of detail from the images, and we finally apply Gaussian filtering to smooth the images, which sometimes contain watermarks and other noisy regions that are useless for our human posture classification task.

The exact operations to compute the HOG features are the following:

- We compute the G_x and G_y gradients of the 60×80 image. We can then compute the magnitude M and the orientation θ of each pixel's gradient by performing: $M = \sqrt{G_x^2 + G_y^2}$ and $\theta = \arctan(G_y/G_x)$
- The image is then divided in small blocks, in our case 8×8 blocks. For each block, we compute the histogram of the gradient orientations θ weighted by their corresponding magnitudes. Each orientation histogram divides the gradient angle range into a fixed number of 9 bins.
- The histograms for each block are then concatenated, creating a single vector that describes the image.
- The vector is then normalized by taking local groups of cells and contrast normalizing their overall responses.

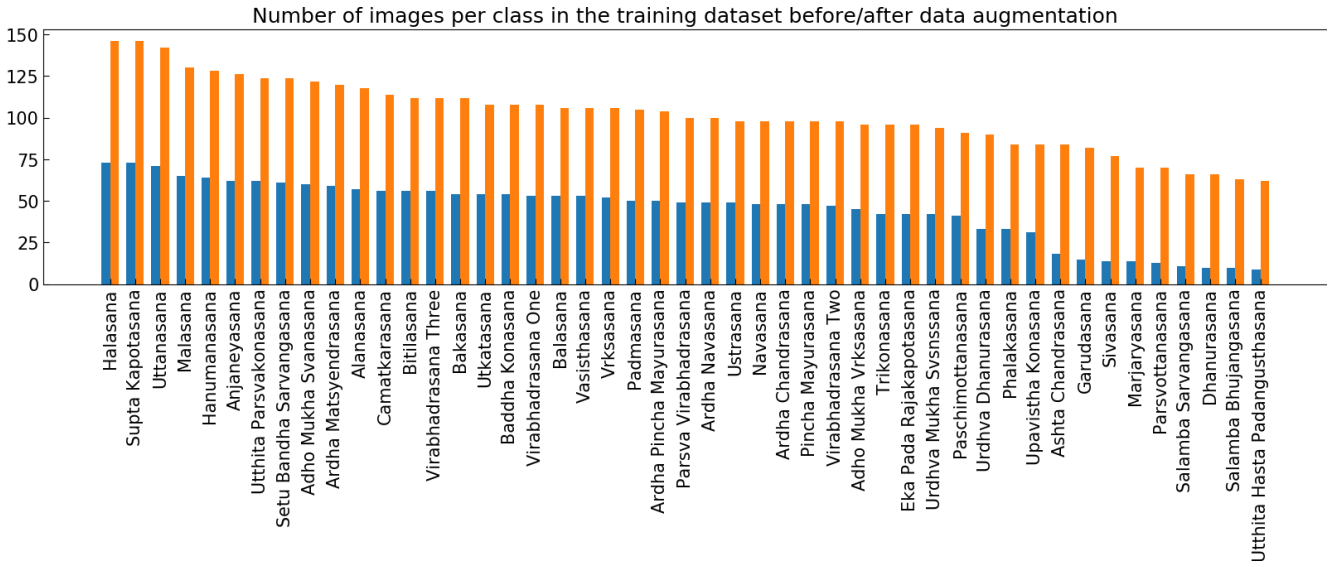


Figure 4. Repartition of the training dataset after data augmentation

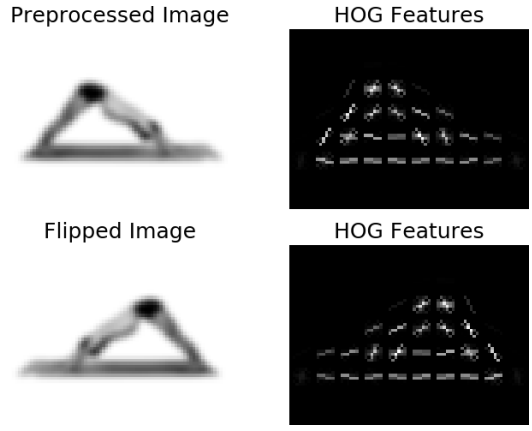


Figure 5. Image and corresponding flipped image along with extracted HOG features from the label *Adho Mukha Svanasana*

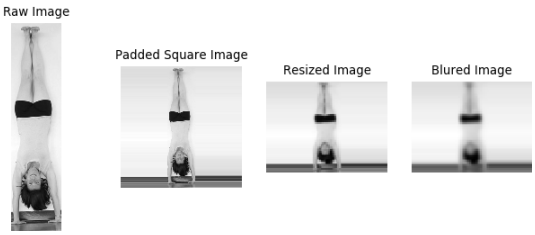


Figure 6. Preprocessing steps on *Adho Mukha Vrksasana* image

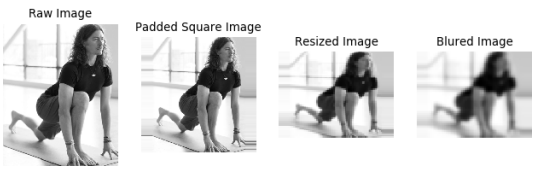


Figure 7. Preprocessing steps steps on *Anjaneyasana* image

Normalization introduces better invariance to illumination, shadowing, and edge contrast. It is performed by accumulating a measure of local histogram "energy" over local groups of cells that we call "blocks". The result is used to normalize each cell in the block.

We use skimage's implementation of HOG features extractor. A visualization of such features is shown in figure 8.

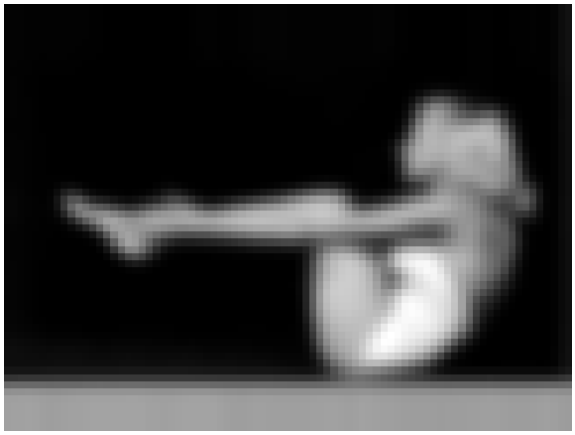
4.4. Classification pipeline

After extracting the HOG features from our images, these features were fed into different classifiers such as RandomForest classifiers and Support Vector Machines. Random Forest is an ensemble learning methods, which means it combines multiple models (decision trees in this case) to improve the overall performance of the model, reducing its variability. On the other hand, SVMs work by finding the hyperplane in a high-dimensional space that maximally separates different classes by maximizing the margin, the distance between the hyperplane and the closest data points from each class.

Combining this different methods in a soft-voting model gives as a robust and rich way of classifying our poses.

4.5. Training

The selection of the best subset of hyperparameters was done by splitting our dataset into three subsets: train, validation and test data. Train and validation datasets were used to test which combination of parameters gave the best results for our different models. This was done using a grid search with 5-Fold cross-validation. Our test dataset was



(a) Original image

(b) HOG features

Figure 8. Image and corresponding extracted HOG features from the label *Ardha Navasana*.

only used when the final models were selected, to compute the estimated error on unseen samples.

4.5.1 Random Forest

The main parameters that were considered in our tests were:

- Number of trees in the forest: this parameter controls the number of trees in the forest. The larger the number of trees, the more accurate the model, but also the longer the training time.
- Maximum depth of each tree: this parameter controls the maximum depth of each tree in the forest. Increasing the maximum depth will make the tree more complex and can lead to overfitting.
- Minimum number of samples required at each leaf node: this parameter controls the minimum number of samples required at each leaf node. A higher value for this parameter will make the model more conservative, reducing the risk of overfitting.

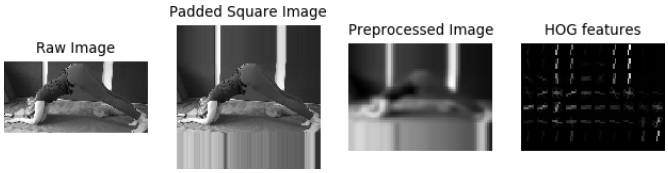


Figure 9. Limitation example of the pre process steps: HOG features are modified by pre process steps on *Ardha Pincha Mayurasana* image

The grid search led to choosing 3000 trees in the forest, with a maximum depth of 26 and a minimum number of samples at leaf nodes of 1.

4.5.2 Support Vector Machines

We selected the radial basis function (RBF) as a kernel for our SVM. The main parameters we decided to explore were:

- Regularization parameter C: this parameter controls the trade-off between maximizing the margin and minimizing the misclassification error. A smaller value of C will result in a larger margin but a larger misclassification error, while a larger value of C will result in a smaller margin but a smaller misclassification error.
- Gamma γ: this parameter controls the width of the RBF kernel. A larger value of gamma will result in a narrower kernel, which can lead to overfitting. A smaller value of gamma will result in a wider kernel, which can lead to underfitting.

The grid search with cross-validation led to choosing $C = 100$ and $\gamma = 0.01$ for our best model.

4.6. Limitations

Limitations of preprocessing The padding step of the preprocessing allows to reduce the deformation due to the resizing (indeed as presented in 4.1. the median size of image is 316x344 and images are resized to 60x80). However, the mode used for padding may add wrong edges to the image, especially when the edge values of the image correspond to a foreground object (as we can see in figure 9, the padded values modify the HOG features).

Limitations due to the background Another limitation of HOG features is happening when the foreground object (here the yoga posture) does not contrast well enough from the background. The HOG features then all have high values. It can be observed in this example. A solution to this problem could be to remove the background, however this task is a complex task itself.

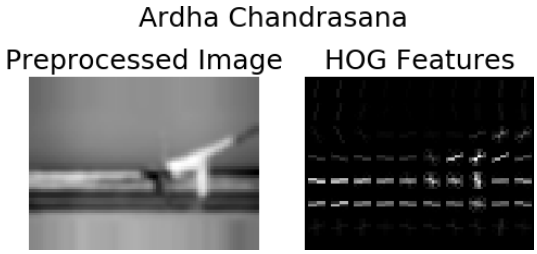


Figure 10. Limitation example: HOG values of the yoga posture are lower than those of the background

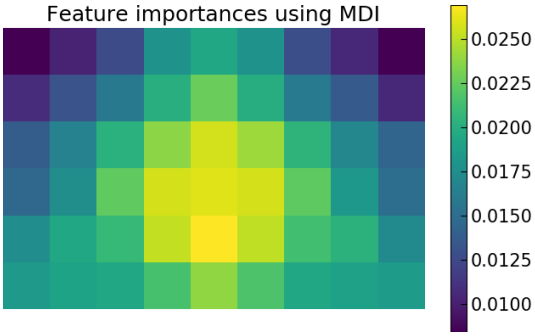


Figure 11. Heatmap of the feature importances per HOG cell

5. Evaluation

5.1. Dimensionality reduction of our features

We tried Principal Component Analysis on our extracted HOG features to check whether the extracted features could be reduced to a smaller manifold of the embedding space. The original space of our extracted features being $\mathbb{R}^{1728}$, we decided to keep 95% of the variance of the data. This reduced our space to $\mathbb{R}^{500}$, which we considered still too highly-dimensional, and came at the cost of losing the interpretability of our visual features since PCA aggregates linear combinations of them, and by losing 5% of accuracy. We therefore decided to continue with our original features to maximize our performance and keep the possibility of visually understanding our classifier.

5.2. Visualization of Feature Importance

In order to check which HOG feature cells the classifier is mostly looking at, we computed the feature importance of the random forest classifier based on mean decrease in impurity. We then sum the importance value per cell of the HOG feature. This results in the feature importance map shown in figure 11.

We observe coherent results, where most important HOG features for the classifier are located at the center of the image and in the bottom corners. On the contrary, the HOG features of the top corners of the images are less useful for classification. This confirms that the classifier is using the human body position and not biased information on the images.

5.3. Metrics for the evaluation of our classifier

The metrics chosen to evaluate the models are:

- F1-score 'macro' (average of the F1-score over the 47 classes without taking into account their size)
- Accuracy

The dataset repartition (3) is imbalanced, yet no class is clearly overriding. In addition, the prediction of one class over another is not critical, therefore neither recall nor precision has to be favoured. As a result, we chose metrics that do not take into account the proportion for each label in the dataset.

The objective of the project was also to compare HOG features and machine learning classifier with neural networks performances. Therefore, we also wanted to have common metrics with Kaggle deep learning competitors. They all published their accuracy score, but only two of them added their F1-score (see 1).

These two reasons (no critical class prediction and metrics comparison) lead us to chose F1-score macro and accuracy metrics.

5.4. Results

The obtained results for each of our classifiers are shown in table 1. As can be seen, soft-voting significantly improves the performance of our models. Models here have been trained on the final splits combining validation and train data and evaluated on previously unseen test dataset. The chosen hyperparameters have been set as mentioned in section 4.5.

We then compare our ensembling classifier to different deep learning methods trained for this particular task in table 2. Our HOG-based classifier outperforms some of the complex architectures that have been pretrained on large datasets of images. This shows that classical computer vision approaches, carefully implemented, can beat off-the-shelf deep learning models. It should be noted the results from deep learning techniques have been retrieved from the original Kaggle notebooks, and are therefore only indicative of the real performance of such methods: the test splits, its sizes, preprocessing and data augmentation techniques are different from one implementation to another, making it difficult to compare the results. These results are therefore only suggesting that our HOG-based classification pipeline is at least as capable as most of the deep learning techniques, allowing much faster inference.

5.5. Error analysis

The obtained confusion matrices for our voting classifiers are shown on figure 12. Data augmentation on the

Method	Accuracy	F1-Score
SVM	69.2	60.8
Random Forest	64.7	54.1
Soft-voting Ensemble	**70.8**	**61.7**
SVM + Data Aug.	75.0	66.7
Random Forest + Data Aug.	71.7	61.3
Soft-voting Ensemble + Data Aug.	**75.7**	**67.2**

Table 1. Results on our test dataset for the different methods implemented.

Method	Accuracy	F1-Score
Xception	61.4	N/A
BiT (BigTransfer)	66.0	N/A
MobileNetV2	68.0	60.0
EfficientNet-B3	90.3*	86.9*
Soft-voting Ensemble	70.8	61.7
Soft-voting Ensemble + Data Aug.	75.7	67.2

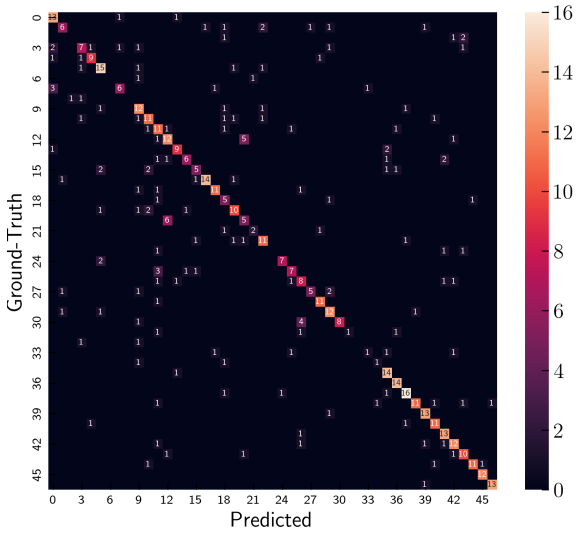
Table 2. Comparison of our classifier to deep-learning based methods tried on this dataset. Experiments with * have used extensive data augmentation techniques to ensure each of the classes contains 100 samples and therefore differ from the other methods.

training dataset clearly helps our model to avoid misclassification errors. In particular, classes 12 and 20, which correspond to *Bitilasana* and *Marjaryasana* postures, were clearly difficult to distinguish for our original model, but this issue has been mainly solved. Note that hog features for these two classes give probably very similar results and classification is probably done through fine details from particular regions in the image like the head or hips from our dataset. Images of both classes are available on figure 13, and show how difficult both poses are to distinguish even for humans. Adding data augmentation to our training dataset ensures that no classes are clearly misclassified and largely improves the final performance of our system.

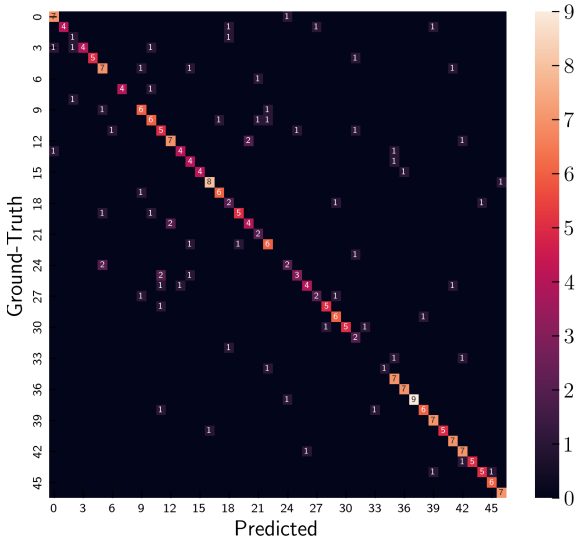
6. Conclusion

To conclude, this project shows that Histogram of Oriented Gradient features are efficient representation of yoga posture in images. Indeed, our best model - a soft-voting ensemble from SVM and random forest using HOG features - achieved an accuracy of 75.7%. On the same task, the first two deep neural networks from the Kaggle dataset achieved an accuracy of respectively 90.3% and 68%. A low computational model with efficient feature extraction can thus achieve comparable results to deep neural networks with high number (>10 millions) of parameters.

In order to improve the model, removing the HOG features located at the top corners of the image could decrease



(a) Original confusion matrix



(b) Confusion matrix after using data augmentation

Figure 12. Confusion matrix of our voting classifier with and without data augmentation. As can be seen, data augmentation clearly helps to reduce misclassification errors.

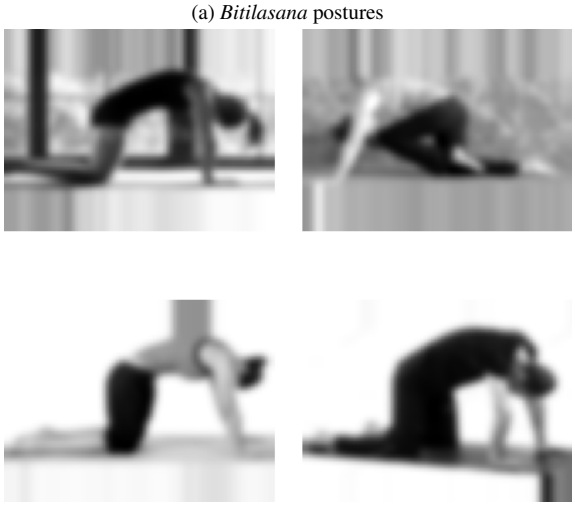
even more the computational cost and improve classification. Also, considering the improvement (+5%) of the accuracy after data augmentation, increasing the dataset (for example by scraping images) can be an interesting investigation.

References

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *Computer Vision – ECCV 2006*, pages 404–417. Springer Berlin Heidelberg, 2006. 2

[2] R. Brunelli and T. Poggiot. Template matching: matched spatial filters and beyond. *Pattern Recognition*, 30(5):751–768, May 1997. 2

(a) *Bitilasana* postures



(b) *Marjaryasana* postures

Figure 13. Examples of hard-to-distinguish images from our dataset. Note that data augmentation clearly helps in classifying this particularly hard to separate labels. In addition, the data we used may contain wrong labels: the bottom-left image from *Bitilasana* seems to be a misclassified example of *Marjaryasana*.

[3] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2016. 1

[4] N. Dalal et B. Triggs. Histograms of oriented gradients for human detection, 2005. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), IEEE, vol. 1, p. 886–893. 1, 2, 3

[5] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning, 2019. 1

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. 2

[7] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004. 2

[8] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. 2019. 1