Action Recognition
Utilizing YGAR Dataset

Shuo Wang, Amiya Ranjan and Lawrence Jiang

DATASCI 281 Computer Vision
UC Berkeley School of Information
{shuo.wang2, aranjan, lawrencejiang0797}@berkeley.edu

Abstract

The scarcity of high quality actions video data is a bottleneck in the research and application of action recognition. Although significant effort has been made in this area, there still exist gaps in the range of available data types in which a more flexible and comprehensive data set could help bridge. In this paper, we present a new process of 3-D actions data simulation engine and generate 3 sets of sample data to demonstrate its current functionalities. With the new data generation process, we demonstrate its applications to image classifications, action recognitions and potential to evolution into a system that would allow the exploration of much more complex action recognition tasks. In order to show off these capabilities, we also train and test a list of commonly used models for image recognition to demonstrate the potential applications and capabilities of the data sets and their generation process.

Introduction

Action recognition is an important area of research within the field of machine learning. Potential applications of movement recognition and understanding of human actions are endless, ranging from robotics and security surveillance to human-machine interactions. Successful progress in this field would translate to solutions for many real-world problems. Not only are the potentials enormous, diverse distinct research topics exist within the broad category of action recognition. Single action classification[1] is one of the more widely studied topics, and shares many similar techniques to image classification [2]. Moving beyond single actions, localized multiple object action recognition are also possible[3, 4]. Even sequential action recognition has been attempted[5]. Although tremendous progress has been made, much still remains to be done. Some of the obstacles facing the research efforts include: relative scarcity of high quality data[1], high resource requirements for training potential model architectures and difficulty in the processing and modification of input data for analysis.

In this paper, we propose a new method of video actions data generation by means of 3D simulation, where data generation could be customized to facilitate various research goals and specific areas of focus, including single action recogition, action orientation detection and action sequence segmentation, to name a few. In addition, we also perform several tests of action recognition using classic image classification modeling techniques and deep learning techniques, demonstrating how our dataset could be leveraged to bridge the gap between image classification tasks and 3D action recognition.

Background

Dataset Several data sets exist currently for single action recognition, including UCF101[6] and HMDB[7]. UCF101 dataset is a collection of youtube videos that contains 101 action categories with a total of 13320 videos, averaging 132 videos per category. Some of the categories in the dataset include "Apply Eye Makeup", "archery" and "Frisbee catch".

HMDB[7] is another data set that aims to further the progress in understanding action recognition. It contains 51 actions and a total of 7000 vidoe clips. The actions within the videos are broadly categorized into five types: general facial actions, facial actions with object manipulation, general body movements, body movements with object interaction and body movements for human interaction.

Over time, larger data sets have been created for video action recognition tasks such as the kinetics dataset[8], where 700 classes of actions have been categoried, each with an average of 926 sample videos.

More recently, spatio-temporally localized atomic visual actions data sets (AVA)[4] have also been introduced, where actions by multiple objects exist within the same video sample.

Action Recognition As more and more data sets come into existence, models have been built to train and test on these data sets, including convolutional neural network, LSTM[1] based encoder models and 3D neural networks that combines frames and optical flow information [1]. More complex architectures for localized action recognitions have also been
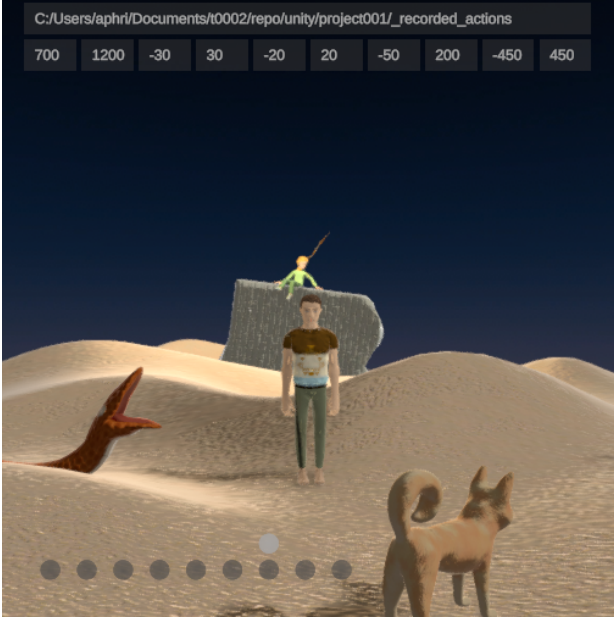
Figure 1: *Action simulation engine.*

proposed[3].

Bottlenecks Although commendable progress has been made in the research of action understanding, many issues still hinder its progress. One of them is the quality of available data sets. The existing data sets are typically collected from video data sources and curated by human judges, this process introduces many variabilities in the quality and comprehensiveness of the sample data, and because these data are collected as they are available, often it is impossible to control for characteristics that are desired for particular research objectiveness. For example, it is often difficult to study the specific effects of object variation and orientation separately. Besides targeted research needs, it is often difficult to study hierarchies of categories. As an example, if we would like to conduct a research where we would like to first recognize the person in the image, and then the action performed by the person, then it would be very difficult to conduct such studies with the existing data sets.

The second issue is the quantity of available data, although kinetics data set has attempted to address the issue of volume with larger number of samples, they are still only single action samples of variable qualities, making studies of localized actions and sequential action recogniton difficult.

We hope that our new data set could help address some of these issues and contribute to the progress of the study in action recognition.

Figure 2: *Various camera angle, offset and zoom configurations.*

Figure 3: *Appearance styles. First set of 4 figures show possible hairstyles. Second set of 4 figures show possible cloth patterns. Third set of 4 figures shows possible pants patterns.*

Data

Data Generation

Our data set is generated by a 3D simulation program(Figure 1) developed in Unity that supports configurations for zoom, center offset, camera angle orientation and avatar styles. The amount of zoom applied to the camera typically varies from 50% to 200%, 100% represents the default zoom amount. An offset could be applied to the x or y direction of the the camera relative to the target of the camera, the unit of this configuration is based on the simulation world space metric system (meters in world space). The camera could also be rotated about the x and y axis relative to the target avatar, we typically set these configurations between -5 degrees to 90 degrees about the x axis and -90 degrees to 90 degrees about the y axis(Figure 2).

The avatar character in the simulation could be configured with various hair, cloth and pants style. In our current iteration, there are 4 different styles for hair, cloth and pants respectively, for a total of 64 unique combinations(Figure 3).

Our first set of actions include a total of 10 different yoga poses: camel, chair, childs, lord of the dance, lotus, thunderbolt, triangle, upward dog, warrior II and warrior III. Each of the 10 yoga poses have 4 type variations within them, some with more pronounced difference than others, for a total of 40 action and action types(Figure 9). We have decided to choose this set of actions as our first action sets because each pose is distinct and has a defined ending position, which allows us to use image classification techniques to compare classic modeling architecture and deep learning ones.

Our simulation also supports the option to include static background and dynamic background objects which would allow us to adjust the complexity of the sample data.
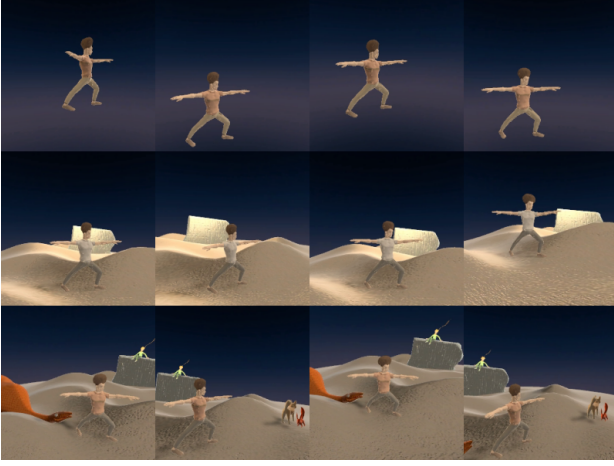
Figure 4: *Difficulty levels. First row shows samples from easy level data set. Second row shows samples from medium level data set. Third row shows samples from hard level data set.*

Table 1: *Data set configurations: zoom in percentage, offset in meters, angle in degrees.*

Type	Easy	Medium	Hard
Min. Zoom	80%	80%	70%
Max. Zoom	110%	110%	120%
Min. X Offset	-1.5	-1.5	-3.0
Max. X Offset	1.5	1.5	3.0
Min. Y Offset	-1.5	-1.5	-2.0
Max. Y Offset	1.5	1.5	2.0
Min. X Angle	-5°	-5°	-5°
Max. X Angle	10°	10°	20°
Min. Y Angle	-30°	-30°	-45°
Max. Y Angle	30°	30°	45°
Static Background	Off	On	On
Dynamic Background	Off	Off	On

Data Set

From the actions and configurations available within our simulation program, we generated 3 sets of video actions data of varying difficulties based on the zoom, offset, angle and scene background configurations specified: easy, medium and hard.

The configurations of the three datasets are listed in the Table 1. Each dataset is generated as follows: for each of the 40 action and action types, we sample 25 random hair, cloth and pants style combinations to create action scenes for, and for each action scene, we capture the action with 20 cameras of randomly generated offsets and angles based on the constraints specified. Each dataset is therefore consisted of 20,000 videos, every action and action type label combination contains 500 videos and every action label contains 2000 videos. Figure 4 shows samples from each difficulty level.
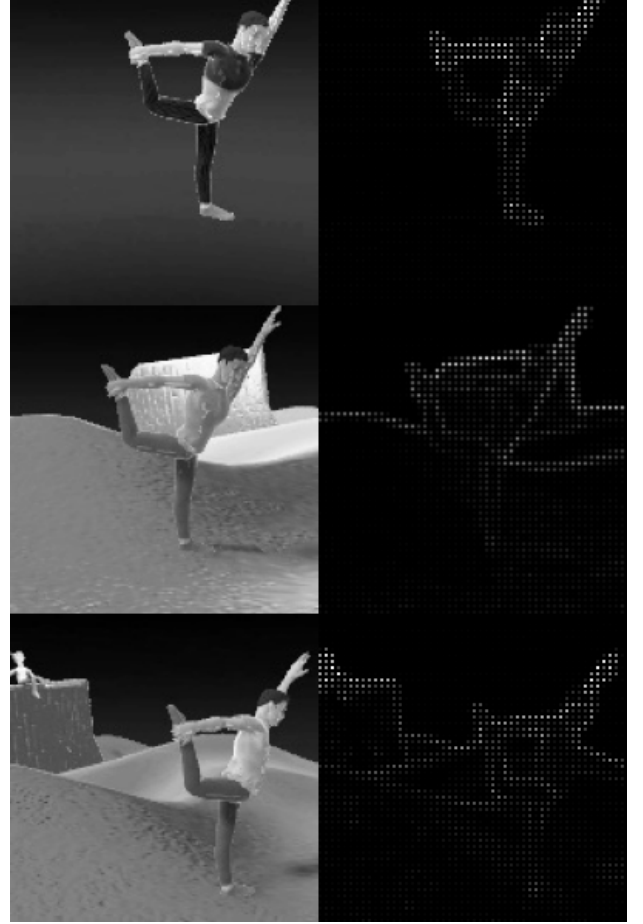
As we can see from Table 1, the configuration of

Figure 5: *HOG filtering of images. First row shows easy level sample, second row shows medium level sample and third row shows hard level sample.*

easy and medium dataset is identical except medium dataset has static background turned on.

Each video is typically about 1 second in length, sampled at 30 FPS. the size of each video is around 80KB and the frame is 351X351 pixels.

Model

Before creating models for the data sets, we have extracted the frame at 50% location of each video as input to our models, since the actions within each video are yoga poses and could be adequately represented by the middle frame. This approach would ideally be enhanced later to use multiple frames within each video, supplemented with temporal information represented by optical flow.

Each frame was extracted from the video as 256X256X3 pixel RBG image, then converted to grayscale image of 256X256 pixels. Afterwards, we cropped top, bottom, left and right of each grayscale image by 70, 30, 50 and 50 pixels to create 156X156 pixels grayscale image, in order to further reduce the size of input data.

Filter

Principal Component Analysis For computing PCA, we first downsized the input grayscale frame by 50% to 78X78 pixels and flatten the image into an array of length 6084. Then we applied PCA on the input array, keeping the top 256 principal components. Finally each of the image array was projected onto the top 256 components to create the PCA weights for each image.

Skeletonize In order to construct image skeletons, we use the thinning algorithm proposed in [9], which uses two iterations of pixel deletion to reduce the image down to a "skeleton" of unitary thickness. Figure 6 shows the before and after results of applying the skeletonize algorithm on the "easy" level data set.

Histogram of Oriented Gradients In order to compute the HOG features, we first cropped the input grayscale image to 100X100 pixels, due to the amount of time HOG computation required. Then HOG feature descriptor was computed with 9 orientations, 2x2 pixels per cell and 2x2 cells per block. Figure 5 shows the results of the filter applied to images of different difficulty levels. From the sample image, it's clear that HOG filtered images are best for images without much structures in the background. As the complexity of the background increases, the HOG features of the avatar become less distinguishable .

Scale-Invariant feature transform & K-Mean Cluster Bag of words is a commonly used technique in image classification. Similar to NLP, image features are used as words. We utilized SIFT features for this purpose. SIFT is one of the important algorithms that detect objects irrelevant to the scale and rotation of the image and the reference. This helps greatly when we are comparing real-world objects to an image because the features extracted are independent of the angle and scale of the image. Figure 7 shows an example of how features from one image could be mapped to another image even when the objects in the image have different orientations. We used OpenCV to extract SIFT descriptors for each image. The descriptors were group into N (N=60) clusters. Then a feature vector $v \in \mathbb{R}^n$ was built where each direction represented a cluster and magnitude represented the count of SIFT descriptors in that cluster for the image.

Classifier

Support Vector Machine We used the support vector classification directly from Scikit-Learn with the default parameters: 1.0 for regularization, radial basis function as kernel and maximum iterations of 50.
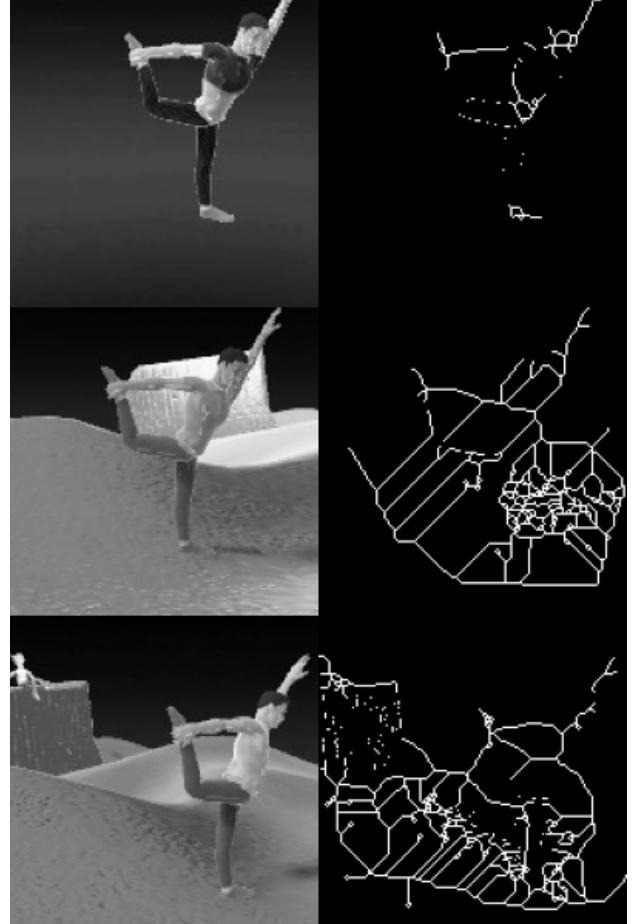

Figure 6: *Skeletonizing of images. First row shows easy level sample, second row shows medium level sample and third row shows hard level sample.*

Logistic Regression Again we used the logistic regression function from Scikit-Learn with default parameters: "LBFGS" as the solver and maximum iterations of 100.

Gradient Boosting Tree The gradient boosting tree we trained used 100 estimators, each with a maximum depth of 3 and a learning rate of 0.1.

Convolutional Neural Network Our convolutional neural network model was inspired by the VGG-16 CNN model. The model consisted of 10 convolutional layers, where the first two layers extracted 32 features and subsequent pairs of layers extracted double the number of features from the previous layers. After each pair of layers, a max pooling of size 2X2 was applied to the input features from the previous convolutional layers. Finally, two dense layers followed by a classification layer were applied to generate the model prediction.
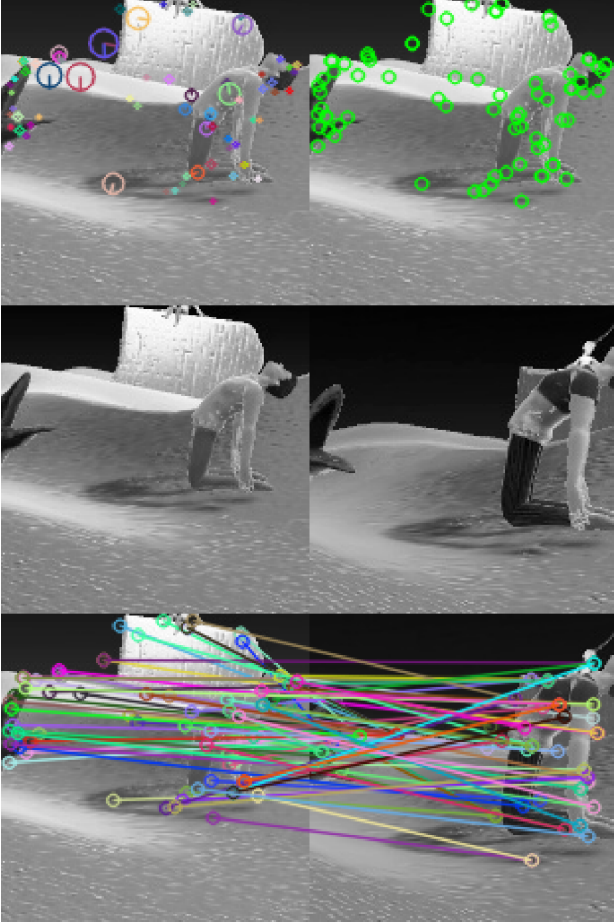
Figure 7: *SIFT feature mapping. First row shows sift features (with and without size) extracted from train image. Second row shows the train image and test image in gray scale. Third row shows the train image features matched to test image features.*

Evaluation

Baseline

Our baseline for evaluation was the SVM model trained on the unfiltered input data scaled down to 20% at the "easy" level. We felt that this model was the most natural starting point to evaluate how effective the various types of filtering and modeling techniques were and how much they contributed to the final accuracy of the models.

Process

We started out by training our baseline models on the "easy" level data set, which produced a guideline for choosing models for further training. Then combinations of filters and models with the best results were trained on the "medium" level data set to assess the capabilities of the models. Finally the best performance models were challenged with the "hard" data set for evaluation.

For every data set and model training, we splitted the sample data into three sets: training set, validation set and test set, in the ratio of 8:1:1. For classic models, validation data set was not used at all, training set was used to train and optionally validate during training depending on the model in question (GBT), and test data set was used to compute the final accuracy of the models.

Results

Table 2 showed the results from training and testing combinations of 4 models and 3 filters. We started out with the easy level data set, training 3 models (SVM, Logistic, GBT) with no filters except downsizing the input images by a further 80%. The results showed that gradient boosting tree generates the best performance out of the three models. However the results of SVM and GBT were comparable with action and action type label combinations. Three more sets of models were trained on the easy level data set with the PCA filter, the HOG plus PCA filter combination and the SIFT filter. The results showed steady improvements, where HOG plus PCA filter with SVM model achieving the best accuracy for action plus action type labels, although the SIFT filtered models performed better on average.

For the medium set, again we performed the same training and testing for all of the classic models, but this time also including the CNN model. As shown in the table, the CNN model easily outperformed all other models, achieving 98% accuracy on the action plus action type labels. Second best performing models were the SIFT feature based models, achieving 28% accuracy on average for action plus action type labels. Looking at the data, a general trend appeared: SIFT filtered models generally performed better. This was not surprising, because one of the key variations in our samples was the orientation of cameras, SIFT features would be able to reduce the complexity due to these variations to a great extent.

Finally, We took the best performing models on medium data set and applied them to the hard data set(retraining on hard data set). The results showed that CNN model still vastly outperformed other models, the SIFT feature models performed moderately well on the action only labels, but faltered when trying to predict the action plus action type labels.

Model Analysis

Hyperparameter Search

Although CNN model produced superior performance by a large margin, The models with SIFT plus

Table 2: *Model results for each difficulty level and model/filter combination. Action column displays results for using action only as label, A+T column displays results for using action and action type combination as label.*

Model	Filter	Easy		Medium		Hard	
		Action	A+T	Action	A+T	Action	A+T
SVM	Downsize	37%	29%	15%	5%	-	-
Logistic	Downsize	43%	16%	21%	6%	-	-
GBT	Downsize	71%	29%	61%	14%	-	-
SVM	PCA	48%	42%	15%	6%	-	-
Logistic	PCA	44%	17%	23%	6%	-	-
GBT	PCA	69%	30%	24%	3%	-	-
SVM	Skeleton|PCA	48%	34%	25%	13%	-	-
Logistic	Skeleton|PCA	47%	16%	35%	10%	-	-
GBT	Skeleton|PCA	72%	25%	41%	7%	-	-
SVM	HOG|PCA	68%	54%	27%	11%	-	-
Logistic	HOG|PCA	55%	21%	34%	10%	-	-
GBT	HOG|PCA	85%	46%	30%	5%	-	-
SVM	SIFT|KMean	50%	43%	34%	28%	16%	9%
Logistic	SIFT|KMean	70%	46%	53%	30%	33%	13%
GBT	SIFT|KMean	71%	44%	56%	27%	32%	12%
CNN	-	-	-	100%	98%	95%	92%

K-Mean clustered features did show great promise. Therefore we performed hyper-parameter search for these models. For this exercise, we used the "hard" level validation data set to optimize the model parameters, Table 3 shows the results. Interesting to note is that SVM performs much better after hyper-parameter search, mainly benefiting from gamma parameter change. Logistic Regression results did not change at all, while GBT results became worse from using a much smaller set of data to estimate the parameters.

T-SNE Visualization of SIFT

Next, we present a t-SNE visualization of the SIFT and K-Mean clustered features of the "easy", "medium" and "hard" data set in order to analyze the distinguishability of the labels within each level. Figure 8 displays the results. Intuitively, we would expect to see the clusters of labels to become less distinguishable as the level of difficulty increases. This is indeed the case as we move from left to right in the figures.

PCA Analysis of HOG

Although we did not apply PCA to the SIFT features, we did observe accuracy improvements applying PCA to the unfiltered image, HOG features and skeleton features. Therefore, we present an exploration of the effect of PCA on the extracted features. Figures 16 and 17 show the eigen images of the PCA decomposition of the "easy" and "medium" data set

with HOG transformation. Looking at the first 16 eigen images of the easy level data set, there is no clearly dicernible patterns within the images themselves, due to the orientation variations in the data set. The eign images of the medium data set, however shows that the background of the images reflect strongly in the eigen images, reducing the accuracy of the features for action identification.

Discussion

Error Analysis

Convolutional Neural Network We performed a detailed analysis of the source of error in the hard level dataset based on the results we obtain from the CNN model. Table 4 shows the accuracy results of the CNN model, which is very high across the board. 10 shows the f1-score of the model across all labels. We see that there are very few variations among them, however, the "triangle" action does see the lowest f1-score. Figure 11 shows the confusion matrix of the "triangle" action labels, which clear indicates that the confusion happens mostly within the different types of the "triangle" action.

Figure 12 shows the incorrect predictions made by the CNN model on the "triangle_3" action and action type combination. Observing these incorrect observations reveal several causes for the errors in prediction. First some angles are difficult for the model to determine what action type is performed as in image 2 of Figure 12. Some of the images simply

Table 3: *Best SIFT | KMean filtered model after hyperparameter search.*

| Model | Filter | Hard | | Parameters |
		Action	A+T	
SVM	SIFT	KMean	40%	16%	C=1 gamma=0.01 kernel=rbf
Logistic	SIFT	KMean	33%	13%	C=0.03 penalty=l2 solver=newton-cg
GBT	SIFT	KMean	28%	10%	learning_rate=0.1 max_depth=2 n_estimators=50

Table 4: *Accuracy of CNN model on hard level data set.*

	Precision	Recall	F1-Score
Macro Avg	93%	93%	93%
Weighted Avg	93%	93%	93%

doesn't contain enough of the avatar to result in an accurate prediction, such as image 7, although the model is still able to predict the action label from just an image of the leg.

SVM, Logistic and GBT Given that the accuracy of the classic models aren't very high on the hard data set, we look at the hyper-parameter searched SVM, Logistic and GBT models with SIFT K-Mean clustered features for error analysis using action plus action type label predictions on the "hard" level data set. Figure 13, 14 and 15 show the confusion matrix of the SVM, logistic and GBT model predictions aggregated to the action labels. We see that the "child" positions are the most accurate category predicted for all three models, while "upward dog" positions are the least accurate, which are very different from the accuracies of the CNN model.

However, considering that the models are still able to classify the samples to action labels with relatively high accuracy, it could be inferred that the models have difficulties recognizing the subtle differences in action types, which is understandable considering the low dimensionalities of the SIFT features.

Our research started out with training three different types of classic image classification models(SVM, Logistic, GBT) in combination with 3 different filtering processes(PCA, HOG, SIFT) using our novel 3-D videos data generation process. The results on the "easy" level data set show that the all three filtering processes significantly improve over the baseline models, demonstrating the abilities of the filters to extract salient features from the images for classifications.

Applying the models on the "medium" level data set shows a more subdued but still respectable model performances on the action labels, yet it is clear that the classic models has reached their limits trying to distinguish between action types. However, SIFT features are able to help the models achieve higher

performance through extracting features that are independent of the background, showcasing its advantages.

Finally, the "hard" level data set results in uniformly poor performance across the classic models, due to increased level of camera orientation difference and additional dynamic background, testing the limits of the SIFT features. Yet the CNN deep learning model is still able to command above 90% accuracy, unequivocally demonstrating the capabilities of the new technique.

Through this process we demontrated how a flexible and comprehensive 3-D data generation process could tremendously improve our research capability and quality, opening doors to many future research topics.

Future

Our work is far from finished. The research results so far is only a starting point, produced in order to showcase what could be possible with the new data generation process. Many topics remain unexplored, some of the immediate next research areas include applying 3D convolutional neural networks on the colored image data, incorporation of optical flow information and applying transformer based model for sequential images embedding and action classification, for both single and sequential actions.

With enhancements to the data simulation engine, we would be able to explore more complex actions and blending of actions. Multiple objects in the scene would allow us to evaluate localized multiple action recognition [3]. Eventually, we could develop a system with the capability to recognize continuous actions for multiple objects within a video data stream.

References

[1] Joao. Carreira et al. Quo vadis, action recognition? a new model and the kinetics dataset. *arXiv:1705.07750*, February 2018.

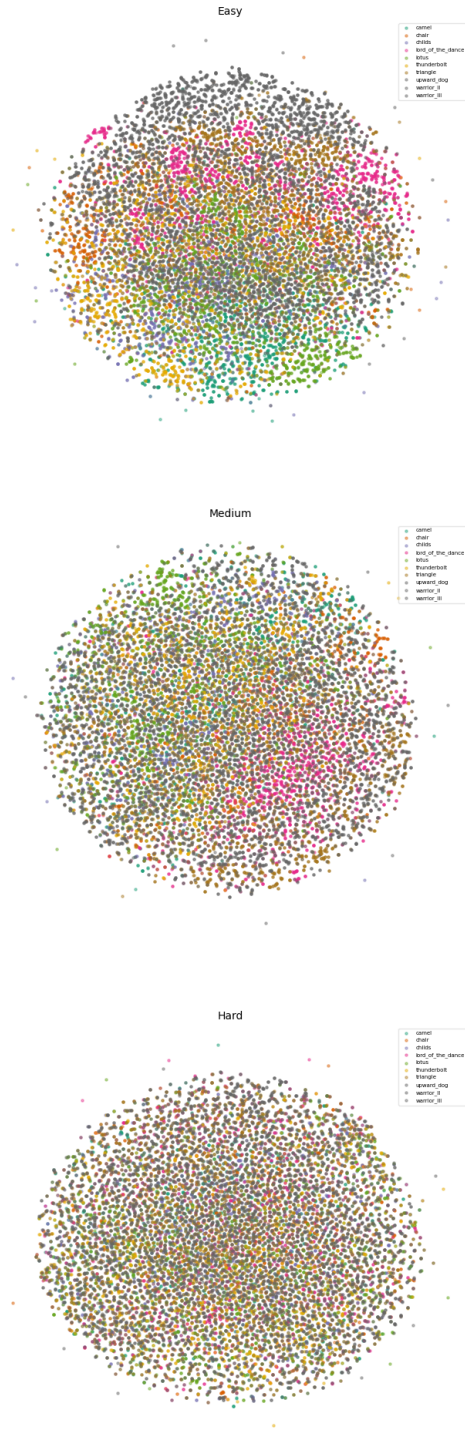[2] Shaoqing. Ren et al. Faster r-cnn: Towards real-time object detection with region proposal net-

works. *Advances in neural information processing systems*, February 2015.

[3] Tao. Wu et al. Stmixer: A one-stage sparse action detector. *arXiv:2303.15879v1 [cs.CV]*, March 2023.

[4] Chunhui. Gu et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. *arXiv:1705.08421v4 [cs.CV]*, April 2018.

[5] Serena. Yeung et al. Every moment counts: Dense detailed labeling of actions in complex videos. *arXiv:1507.05738v3 [cs.CV]*, April 2018.

[6] Khurram. Soomro et al. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv:1507.05738v3 [cs.CV]*, November 2012.

[7] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[8] Lucas. Smaira et al. A short note on the kinetics-700-2020 human action dataset. *arXiv:2010.10864v1 [cs.CV]*, October 2020.

[9] T. Y. Zhang. et al. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, *Volume 27*, *Number 3*, March 1984.

Figure 8: *t-SNE visualization of SIFT plus K-Mean clustered features of "easy", "medium" and "hard" level data sets.*

APPENDIX

Action Figures

Figure 9: *10 yoga poses, each with 4 different variants.*
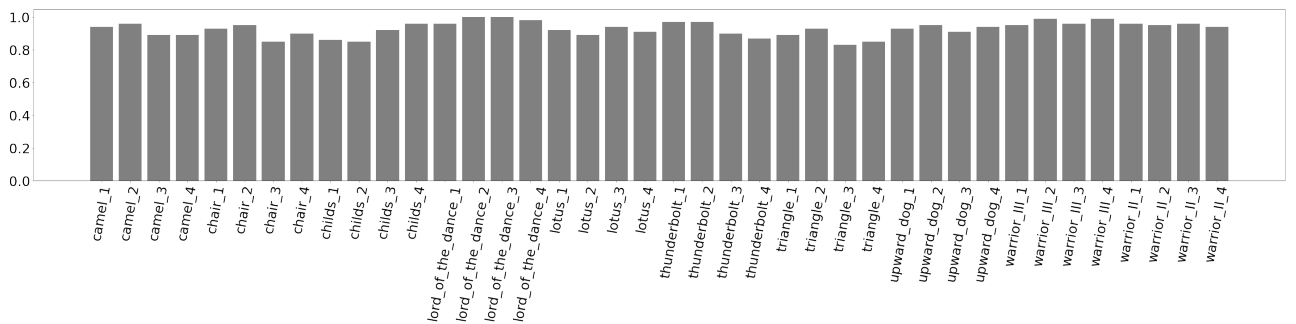
Model Errors and Results

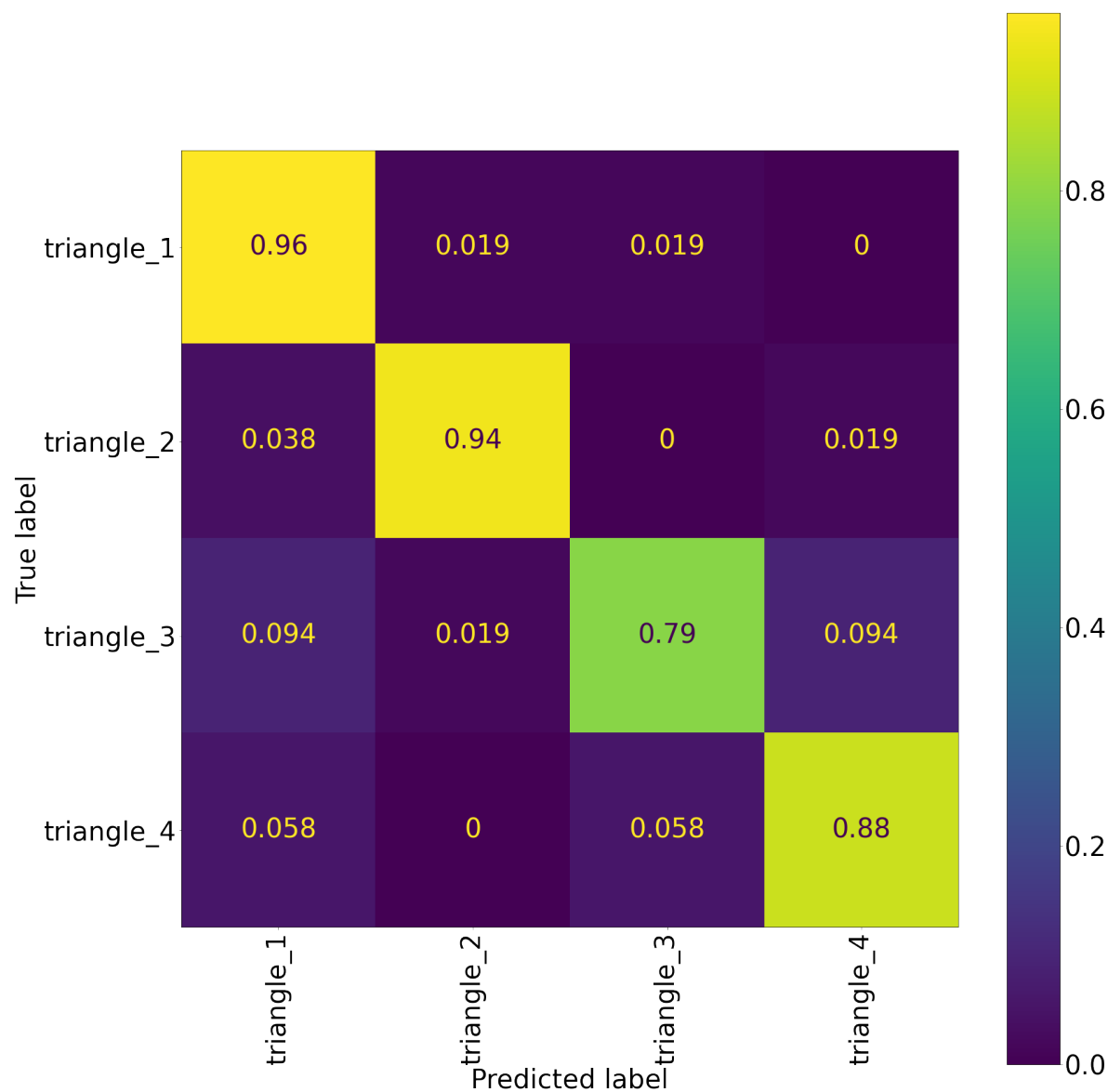

Figure 10: *F1 scores of the CNN model across labels.*

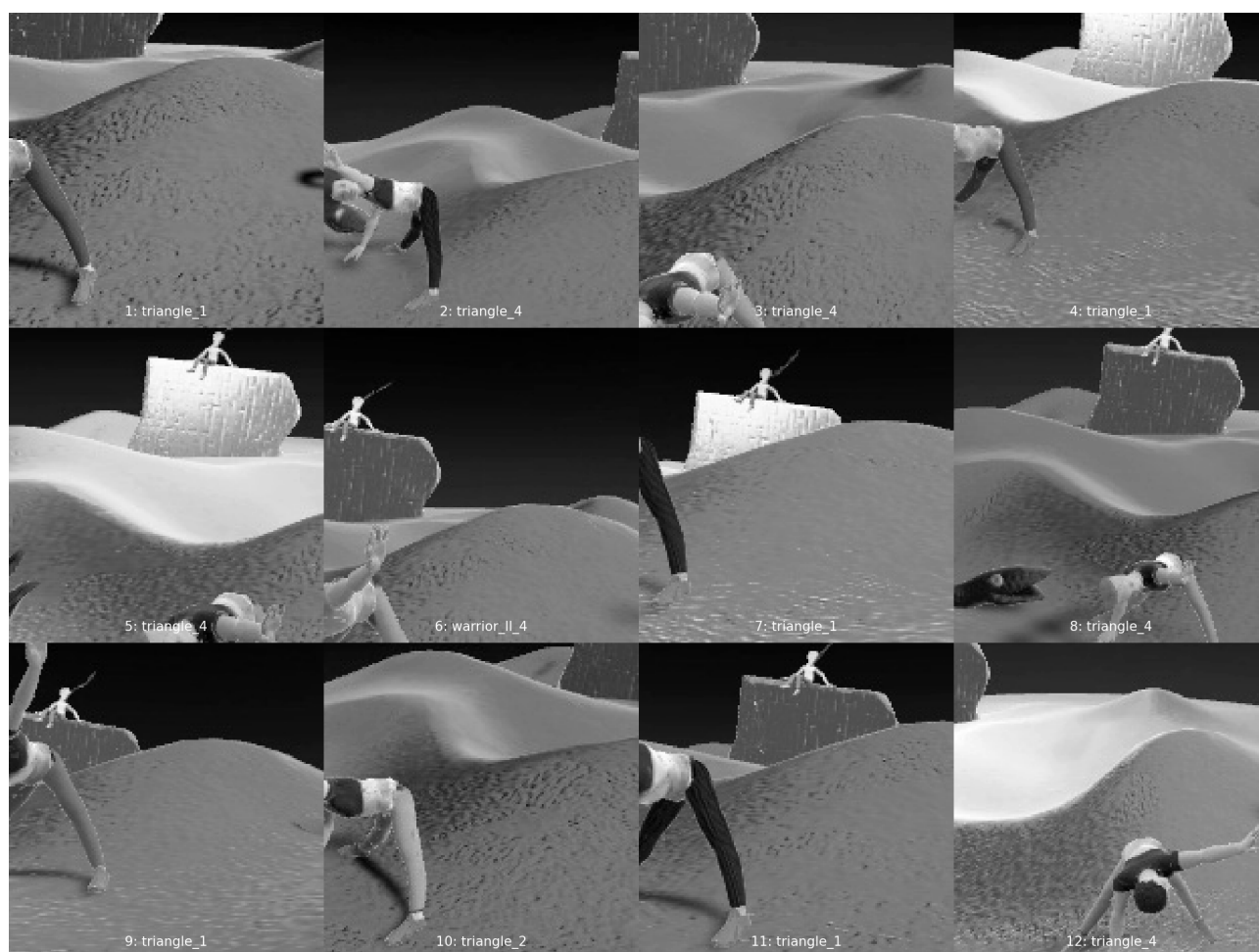**Figure 11:** *CNN model confusion across "triangle" action labels.*

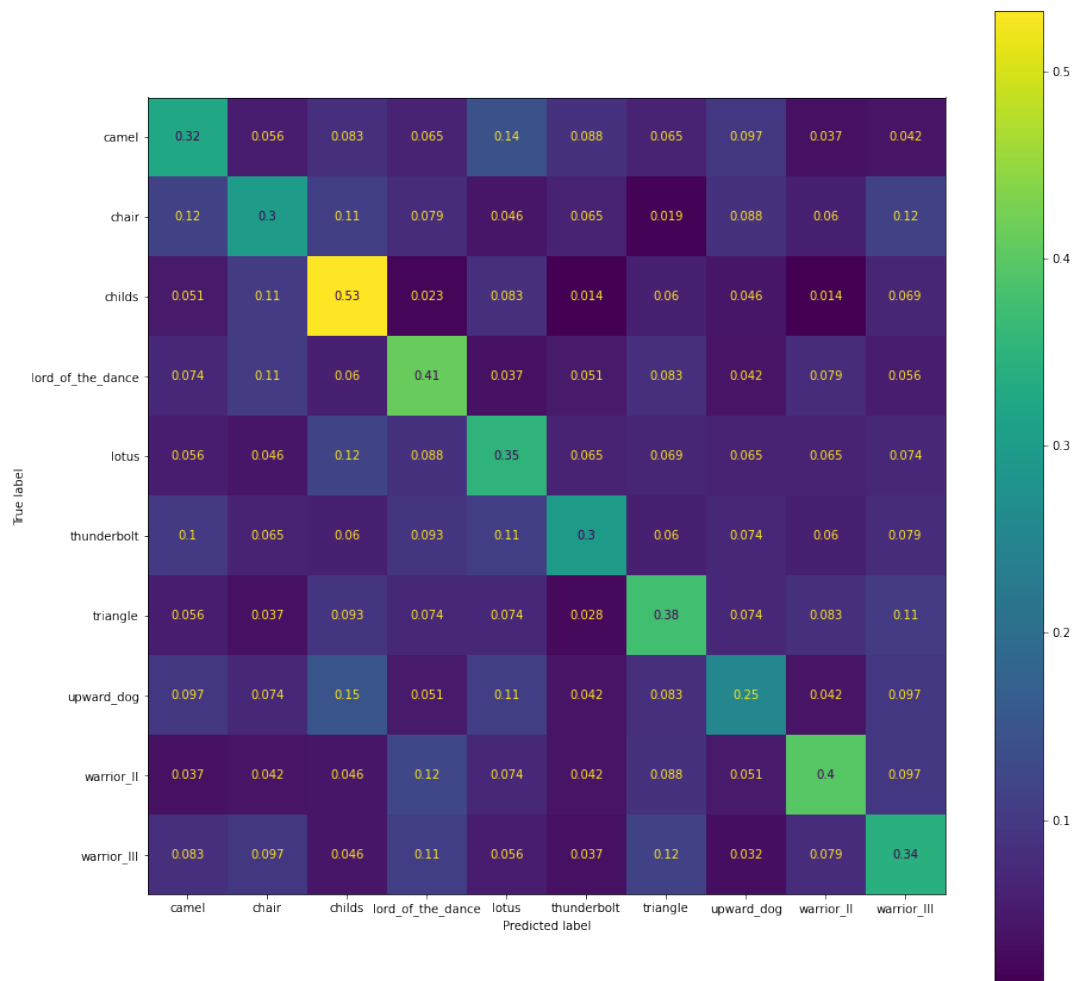**Figure 12:** *CNN model confusion across "triangle" action labels.*

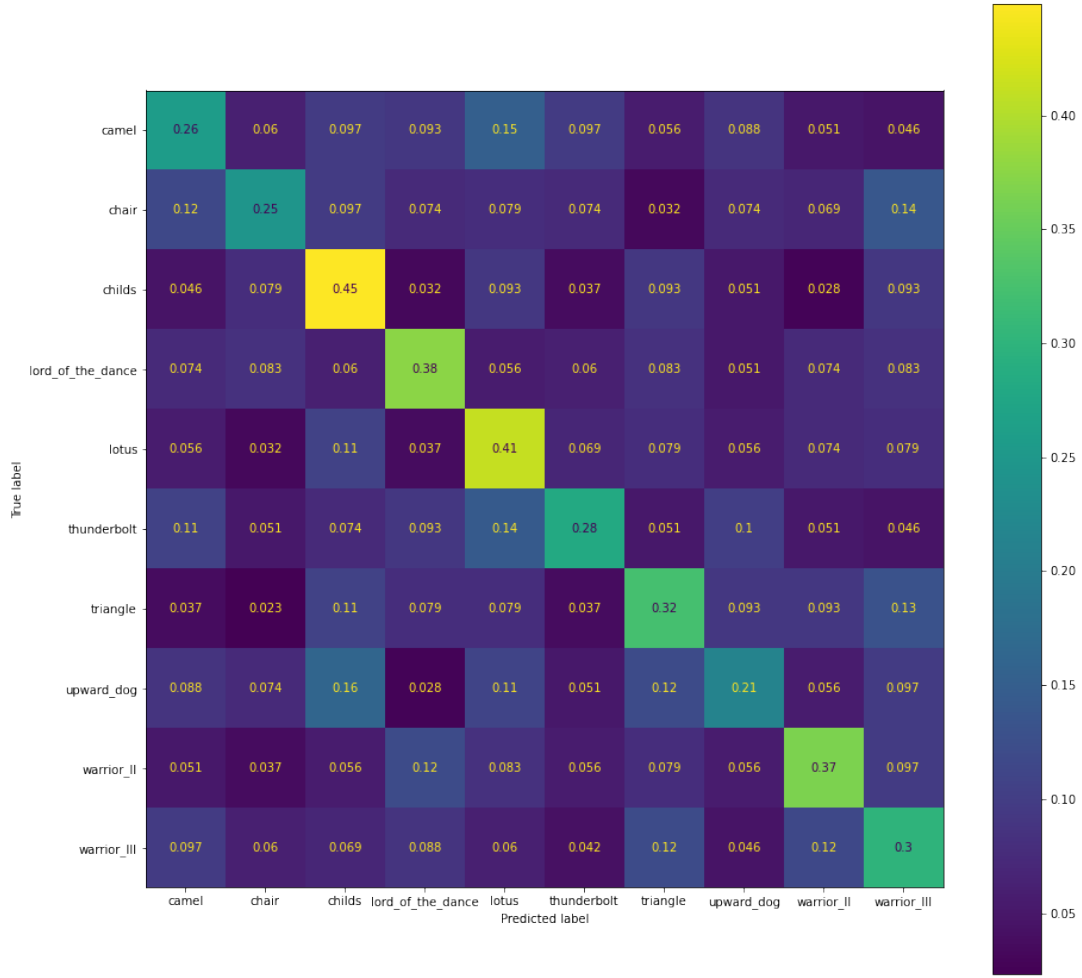**Figure 13:** *SVM model confusion matrix aggregated to the action labels.*

[H]

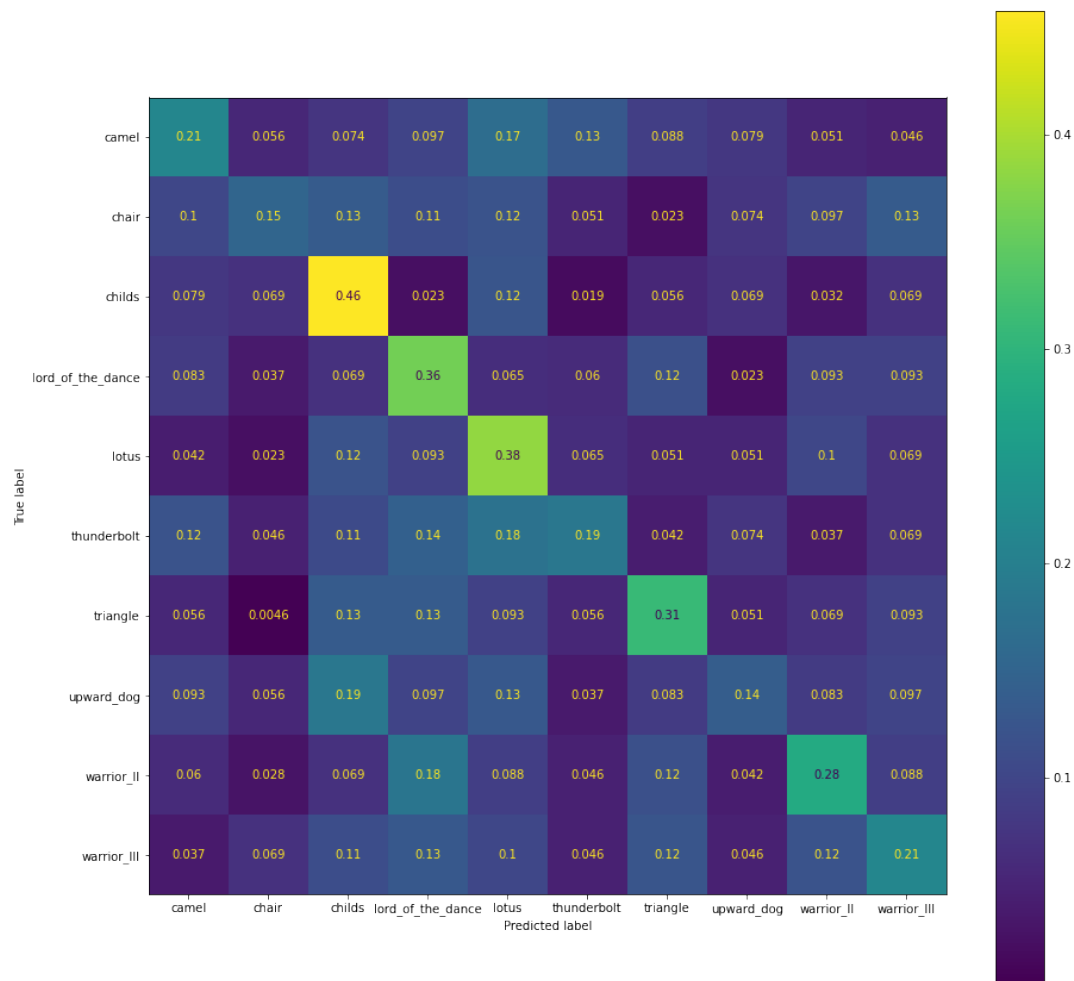**Figure 14:** *Logistic model confusion matrix aggregated to the action labels.*

Figure 15: *GBT model confusion matrix aggregated to the action labels.*
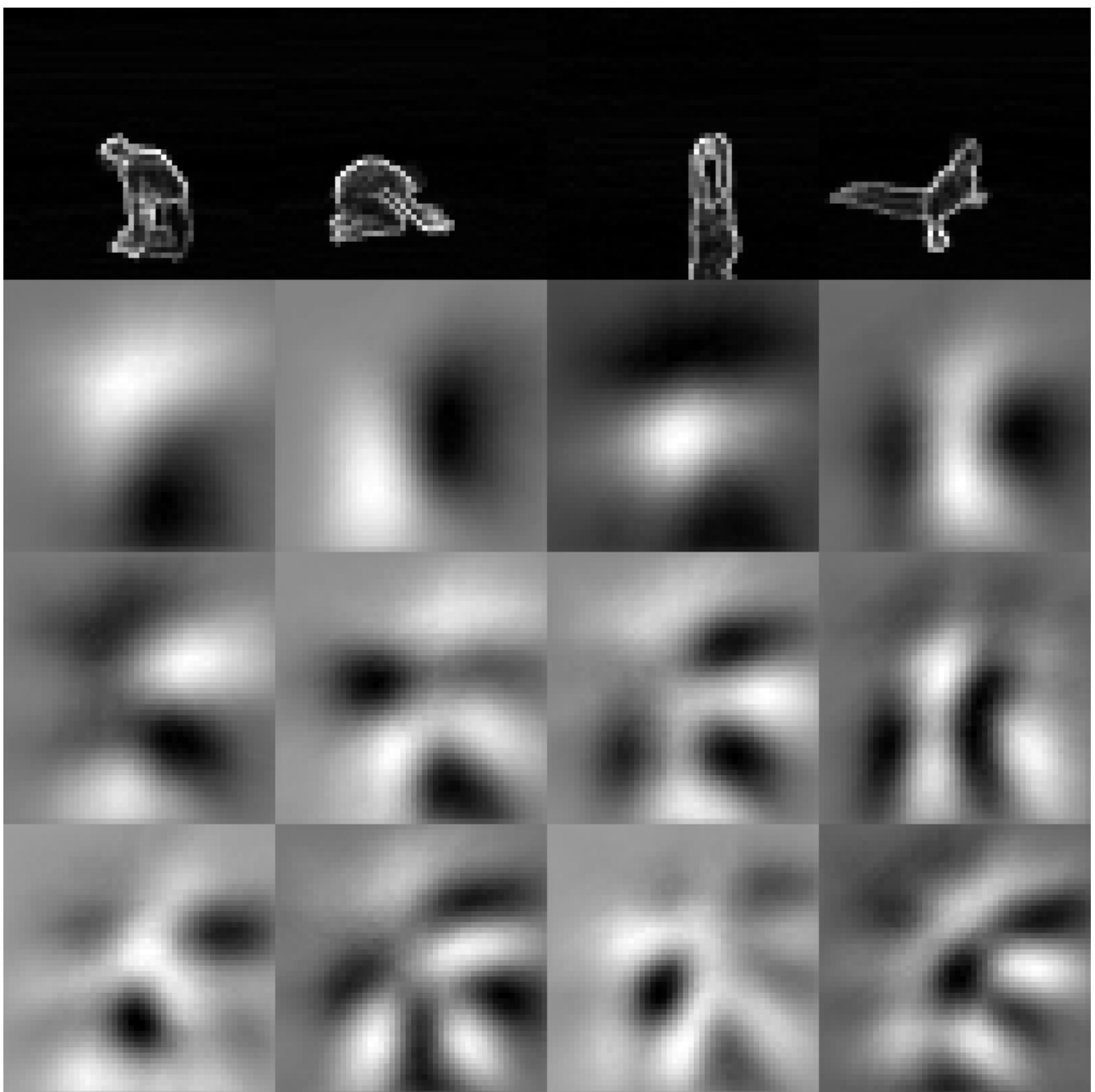
PCA Decomposition of HOG Features

Figure 16: *PCA decomposition of easy level data set with HOG filter. First row contains sample HOG images from the data set.*
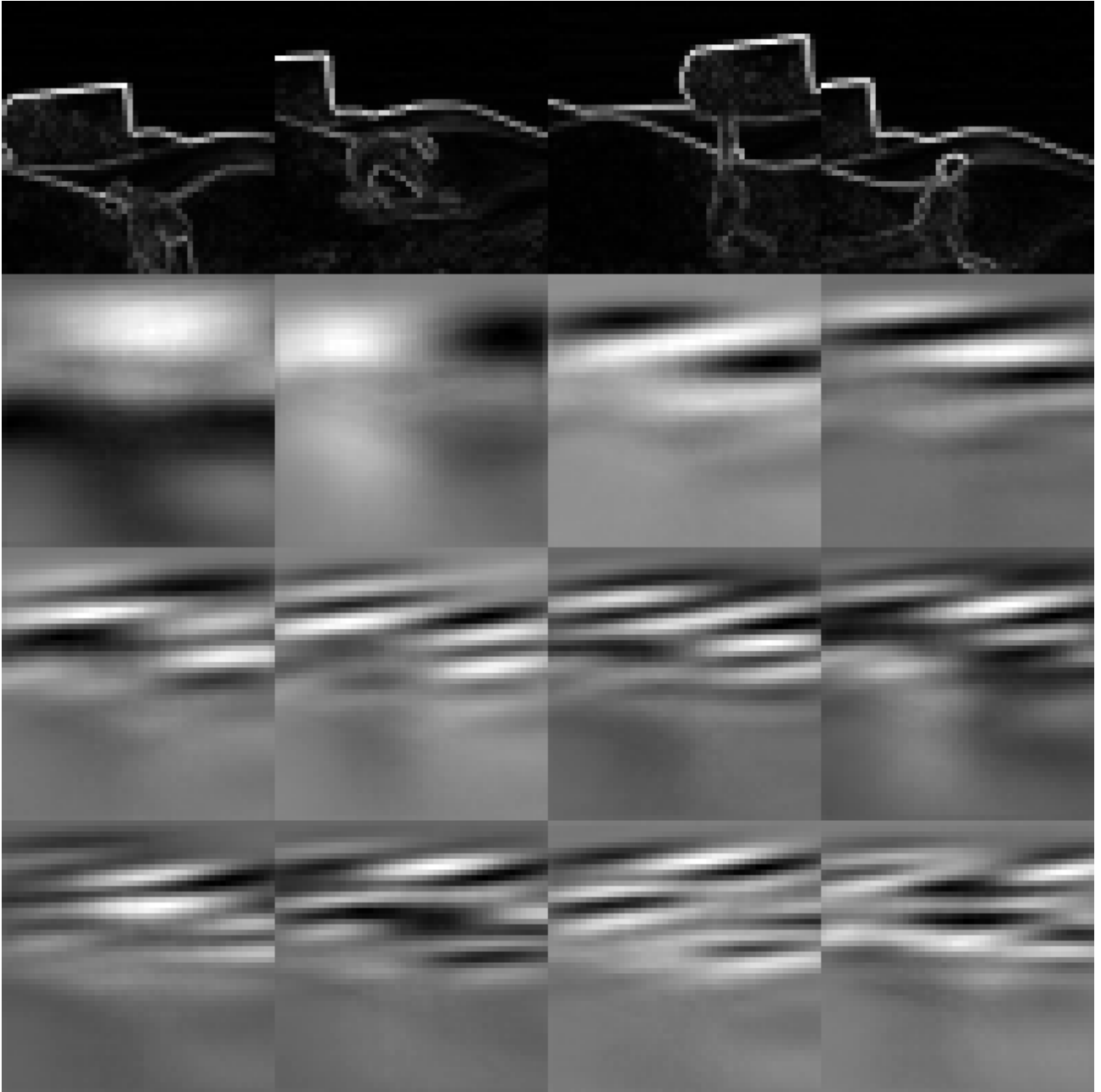
Figure 17: *PCA decomposition of medium level data set with HOG filter. First row contains sample HOG images from the data set.*