Action Recognition

Utilizing YGAR Dataset

Shuo Wang
Amiya Ranjan
Lawrence Jiang

Berkeley
UNIVERSITY OF CALIFORNIA

Motivation & Literature Review

- 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.
- Existing data include: UCF101 and Kinetics, etc.
- We propose a new method of customizable video actions data generation by means of 3D simulation, with the goal of generating high quality AR data easy for targeted AR research.
- We also perform several tests of action recognition using classic image classification modeling techniques and deep learning techniques.

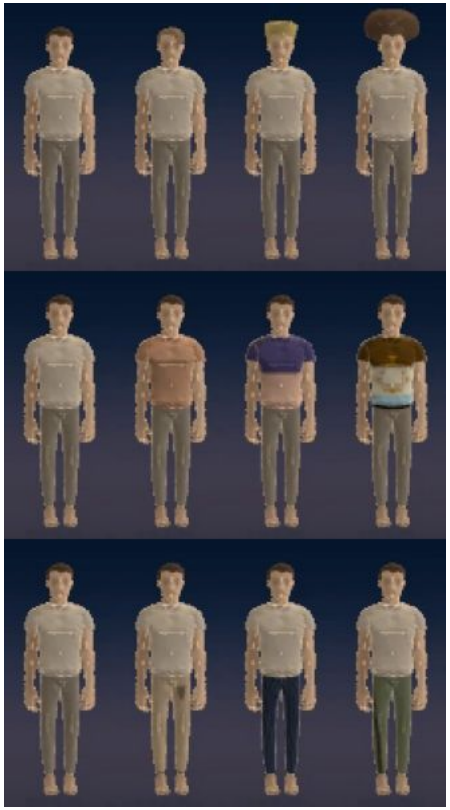
Problem Statement

- Our data set is generated by a 3D simulation program developed in Unity that supports configurations for zoom, center offset, camera angle orientation and avatar styles.
- The actions in the data contains 10 yoga poses, each pose contains 4 varieties. The goal is to train models to recognize these actions.
- 3 sets of data, each action type, 25 styles, 20 camera orientations
- Use middle frame as image data input

Data Generation

Data Generation - continued

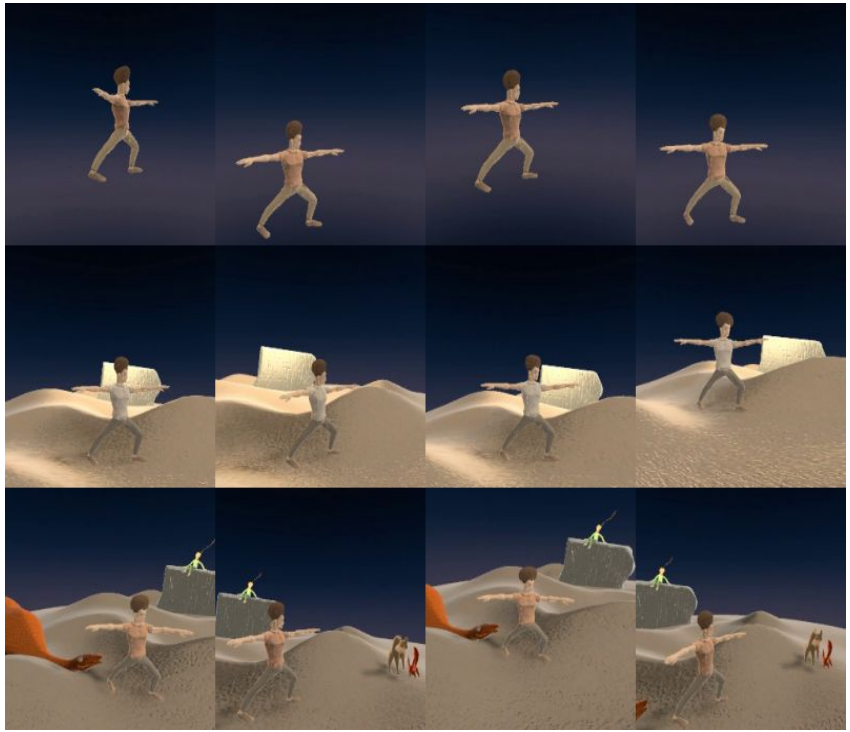
- Our data set is generated by a 3D simulation program developed in Unity that supports configurations for zoom, center offset, camera angle orientation and avatar styles.



Data Set: Easy, Medium and Hard

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



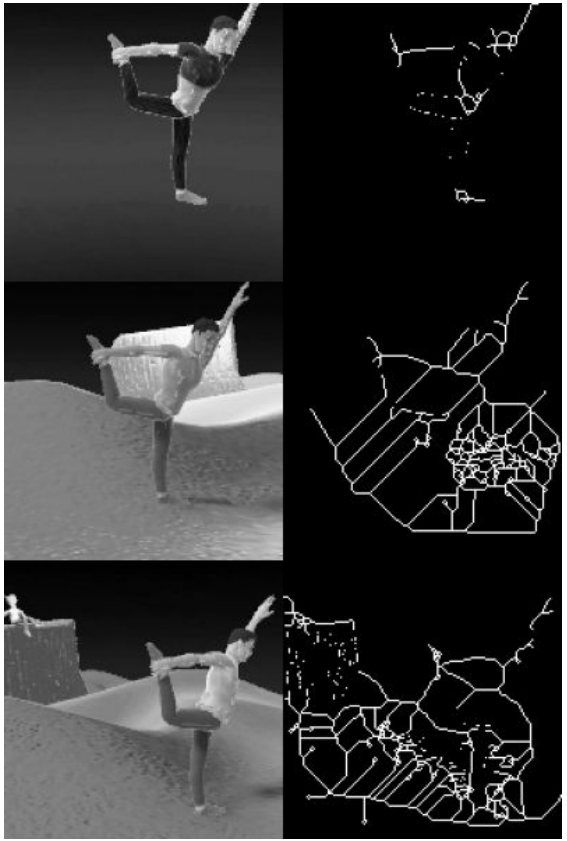
Proposed Methods

- Filters
- Models
- Experiment

Filter: Skeleton + PCA

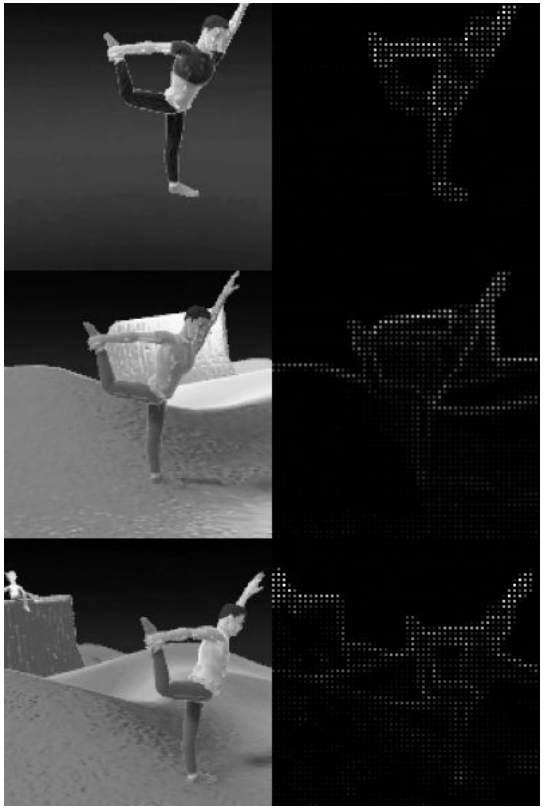
Skeleton: uses iterations of pixel deletion to reduce the image down to a “skeleton” of unitary thickness.

PCA: Apply PCA on the input image, keeping the top 256 principal component. each image sample projected onto the components to create the PCA weights



Filter: HOG + PCA

- HOG: crop the input grayscale image to 100X100 pixels, HOG feature descriptor was computed with 9 orientations, 2x2 pixels per cell and 2x2 cells per block.
-
- PCA: Apply PCA on the input image, keeping the top 256 principal component. each image sample projected onto the components to create the PCA weights



Filter: SIFT + KMean

- SIFT is an important algorithm that detects objects irrelevant to the scale and rotation of the image and the reference.
- 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. The descriptors were group into N (N=60) clusters.
- Reducing cluster number reduces power of prediction/ increasing requires much higher computing power.

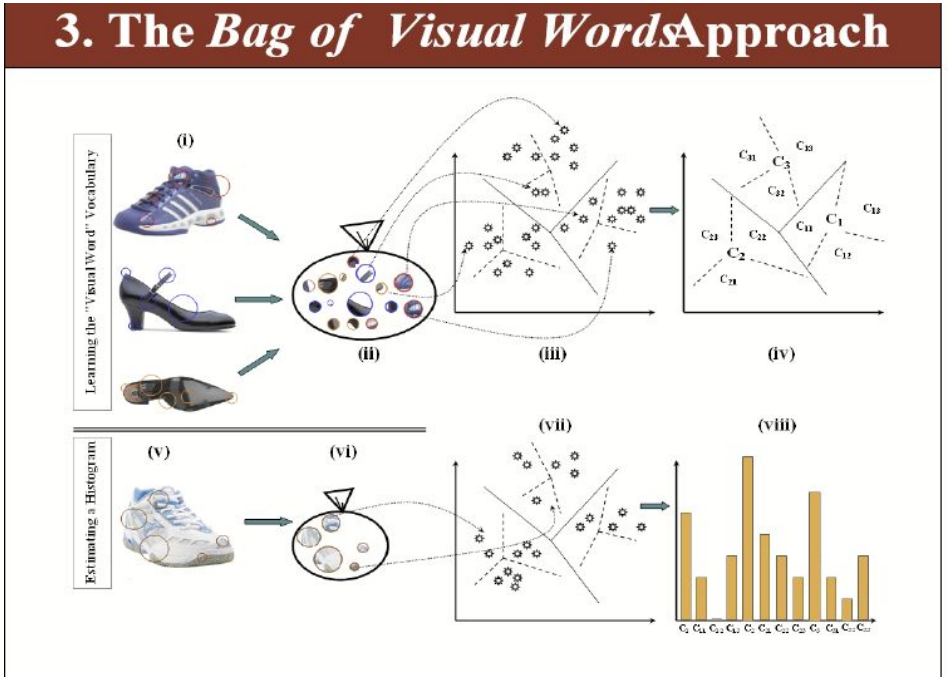
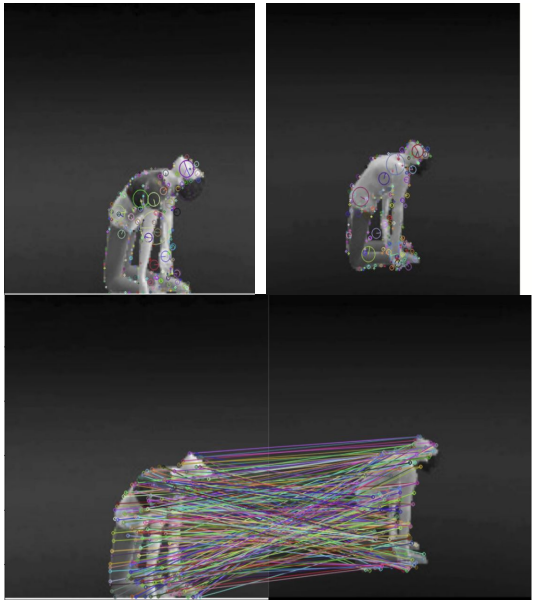
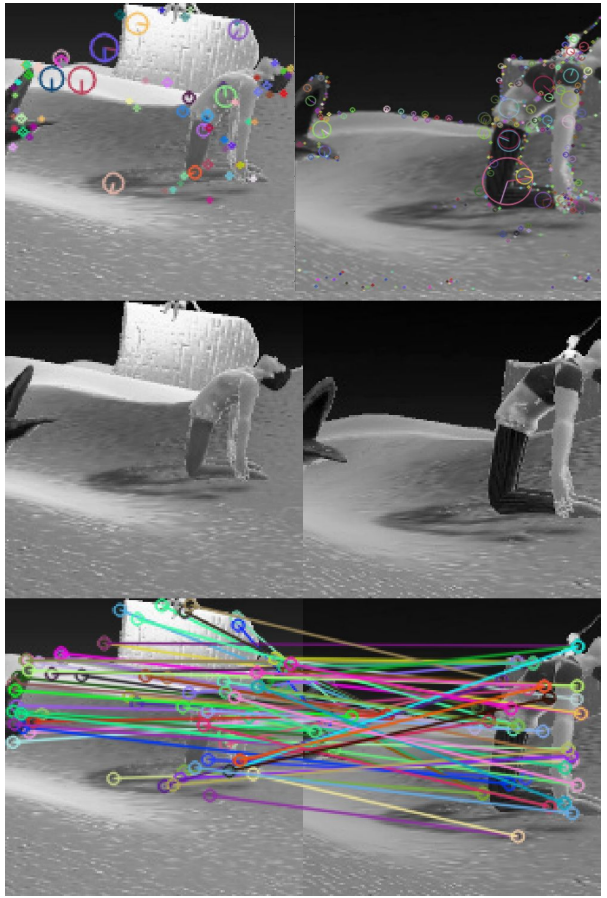


Figure 1. Bag of Word approach. Adapted from "Tagging Products for image classification " by B. Tomasik et al. Retrieved from https://briantomasik.com/tagging-products-using-image-classification/

Filter: SIFT + KMean





Classifiers: Classic

- SVM: 1.0 for regularization, radial basis function as kernel and maximum iterations of 50.
- Logistic: “LBFGS” as the solver and maximum iterations of 100.
- GBT: 100 estimators, each with a maximum depth of 3 and a learning rate of 0.1.

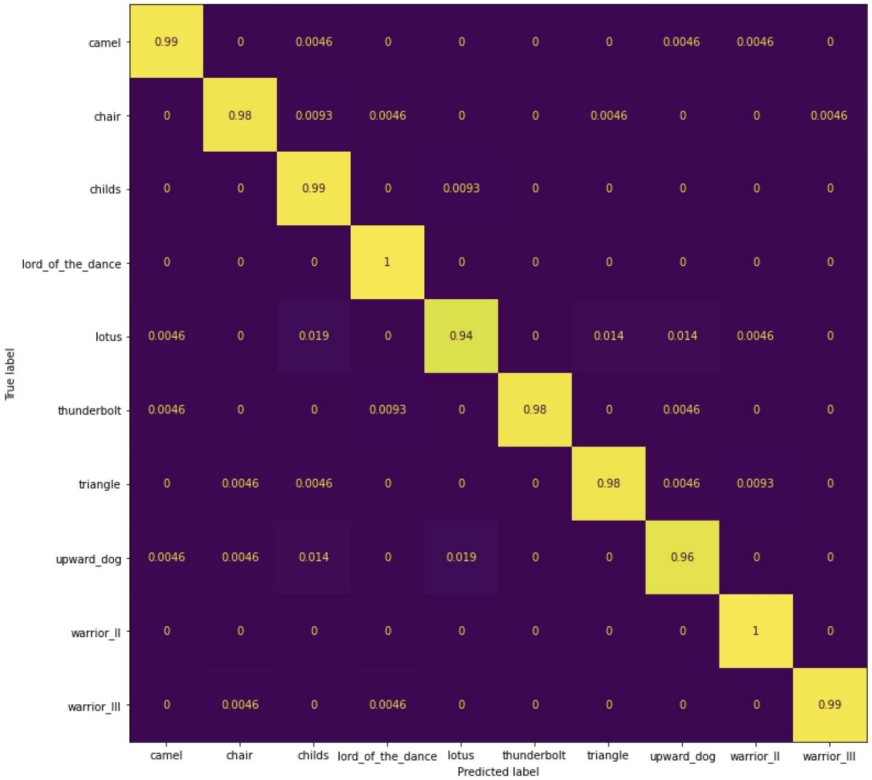
Deep: Convolutional Neural Network

- 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.

Experiment: Train & Test Process

- Our baseline for evaluation was the classic models trained on the unfiltered input data scaled down to 20% at the ``easy'' level.
- Combinations of filters and models were trained on the “Easy”, “Medium” and “Hard” data set.
- We split the sample data into three sets: training set, validation set and test set, in the ratio of 8:1:1.

Analysis: CNN



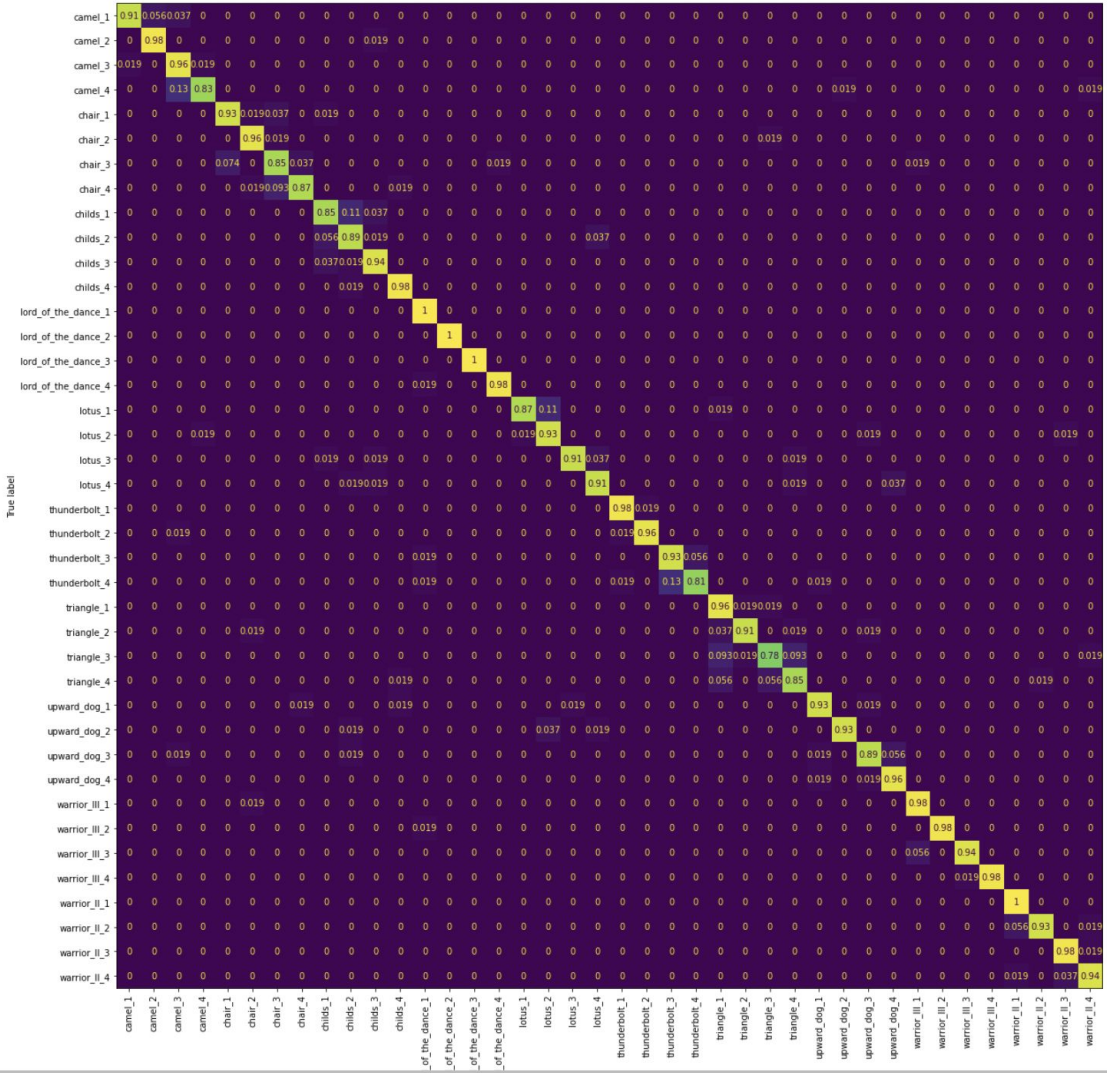
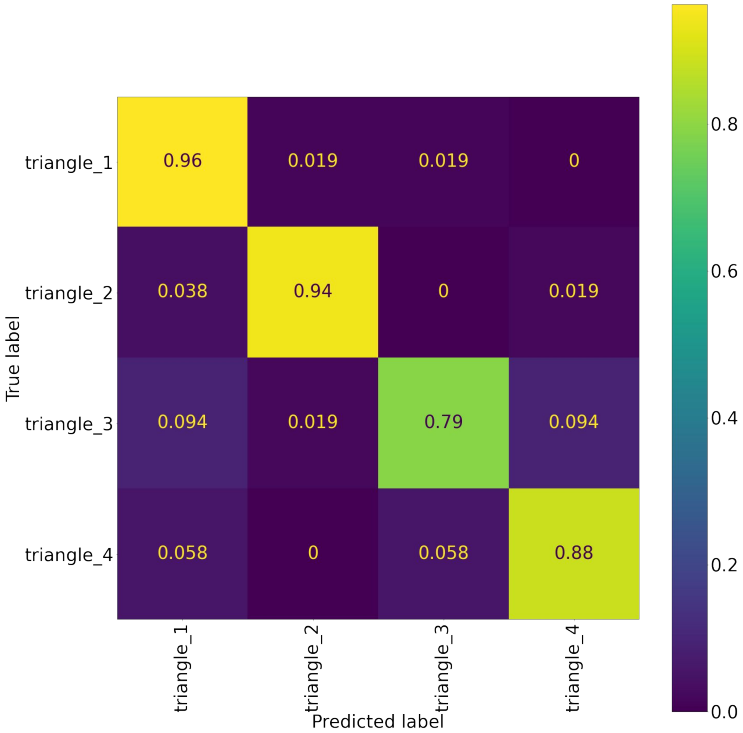
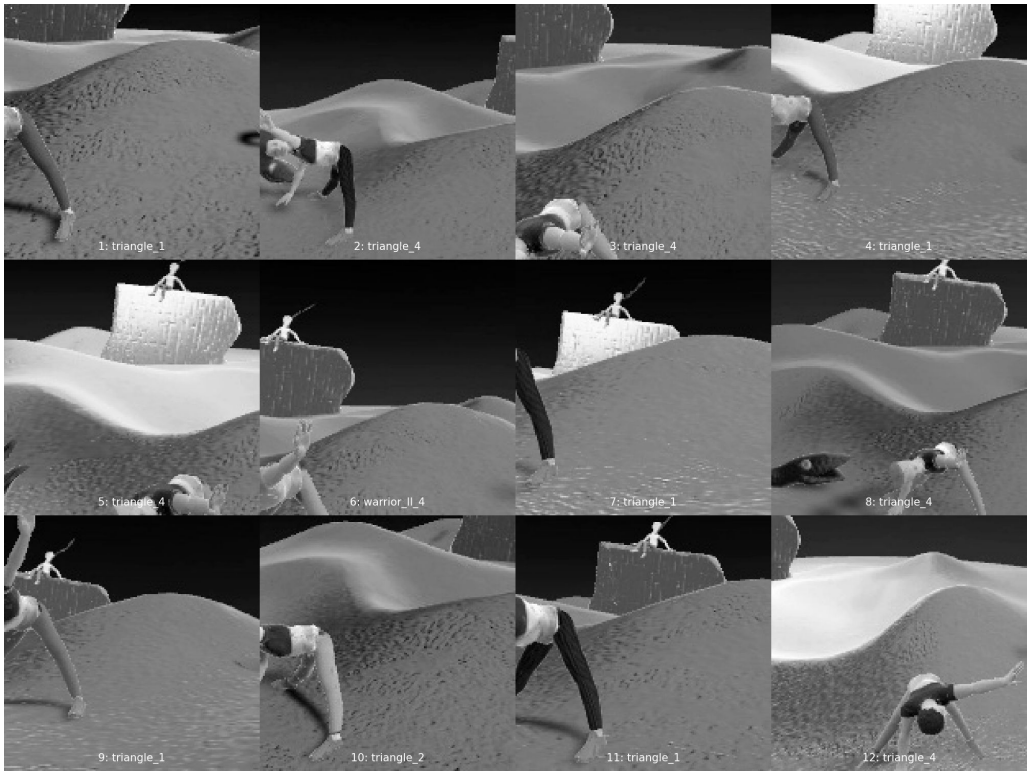


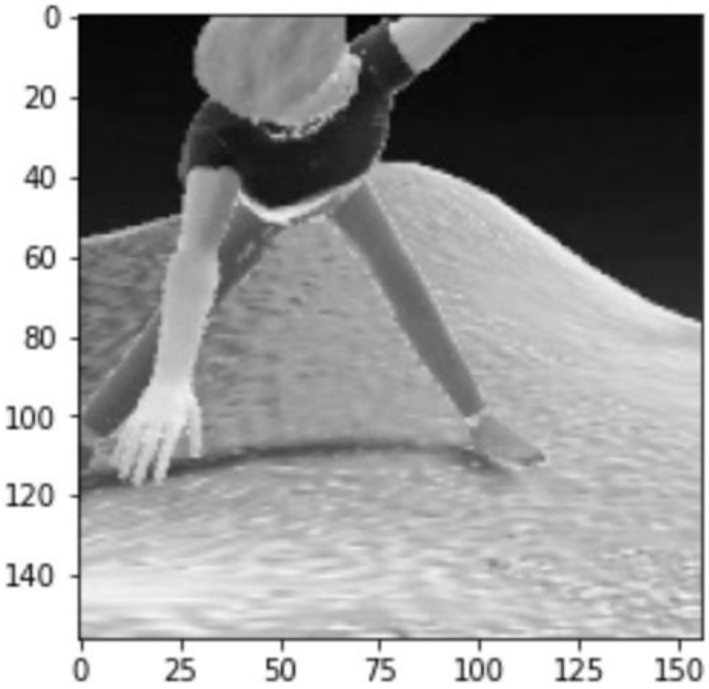
Table 4: *Accuracy of CNN model on hard level data set.*

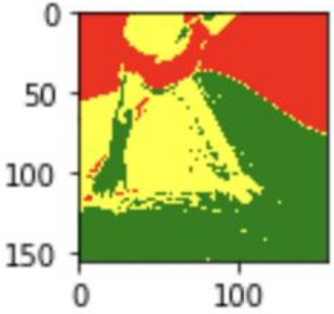
	Precision	Recall	F1-Score
Macro Avg	93%	93%	93%
Weighted Avg	93%	93%	93%

Analysis: CNN - continued






0
20
40
60
80
100
120
140
0
25
50
75
100
125
150


0
50
100
150
0
100

Results

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.*

		Easy		Medium		Hard	
Model	Filter	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%

Analysis: Classic Models Hyper-Parameter Search

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	33%	10%	learning_rate=0.1 max_depth=2 n_estimators=50

Discussion

- The results on the “easy'” level data set show that 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.
- On “medium” level, SIFT features are able to help the models achieve higher performance through extracting features that are independent of the background, showcasing its advantages.
- On “hard” level, the CNN deep learning model is able to achieve above 90% accuracy.

Future

- apply 3D convolutional neural networks on the colored image data, incorporation of optical flow information.
- apply transformer based model for sequential images embedding and action classification, for both single and sequential actions.
- enhancements to the data simulation engine, explore more complex actions and blending of actions.
- multiple objects in the scene would allow us to evaluate localized multiple action recognition.
- develop a system with the capability to recognize continuous actions for multiple objects within a video data stream.
- Add capability for the simulation engine to calculate percentage of avatar in camera.
- More comprehensive hyperparameter search for GBT
- Apply model Pre-trained on our data to public data sets (such as ucf101) and test the transfer learning potentials of the system