Penn Engineering
University of Pennsylvania

Evaluating Pose Similarity in TikTok Videos

CIS 581 Final Project
Fall 2020

Alex Chen, James Rao, Jonathan Lee, Mona Lee, and Tatiana Tsygankova

Goal

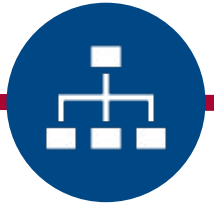
Given a target dance video and a second dance video, determine a score (1-10) for how accurate the second video is at replicating the dance in the target video.

Roadmap

Data Pre-Processing

- Training Data Preparation
- OpenPose
- Feature Refinement

Model Architecture



- Pytorch Neural Network
- Activation Function Experimentation
- Parameter Tuning

Results and Discussion

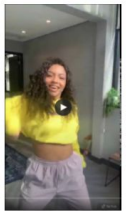
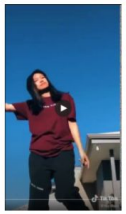
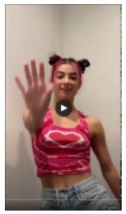
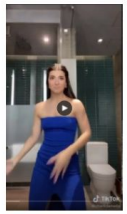
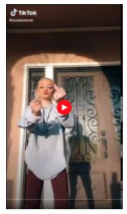
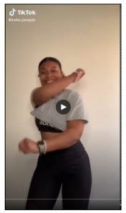
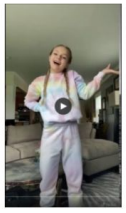
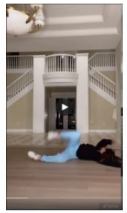
- Findings
- Discussion
- Next Steps

Data Pre-Processing

Training Data Preparation and Extracting Features from OpenPose

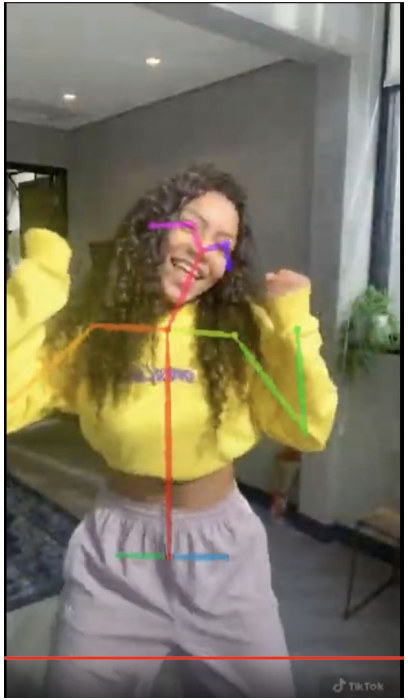
Collecting Data

- **12** dances
- **15-25** videos per dance
- Team scored each video 1-10 based on labeled target video

All About Cake	Corvette	Don't Start Now	Elastic	Get Busy	Get Up
					
Renegade	**Savage**	**Say So**	**The Git Up**	**THICK**	**WAP**
					

Pre-Processing

- OpenPose to output body keypoints for each frame (30 fps) for each video
- Normalize keypoints to [0, 1] scale
- Centered frame around the nose by setting its coordinate values to (0.5, 0.5)



The Model

Neural Network for Multiclass Classification

Initial Approaches

Euclidean distance, cosine similarity, various model architectures

- We experimented with a variety of scoring techniques, finding that the most robust way to calculate similarity was using a neural network
- We also conducted parameter tuning: searching for the optimal number of hidden layers, presence of a Dropout layer, and various activation/loss functions.

```
def compute_cosine_similarity(filepath1, filepath2):
  frame1Keypoints = []
  frame2Keypoints = []
  with open(filepath1) as frame1:
      itemData = json.load(frame1)
      frame1Keypoints = itemData['people'][0]['pose_keypoints_2d']
  with open(filepath2) as frame2:
      itemData = json.load(frame2)
      frame2Keypoints = itemData['people'][0]['pose_keypoints_2d']

  #get xy vector for frame1 (drop the confidence from the keypoint data)
  frame1_xy = get_xy_coord_array(frame1Keypoints)

  #frame2 xy vector
  frame2_xy = get_xy_coord_array(frame2Keypoints)

  #cosine similarity from scipy
  result = cosine_similarity(frame1_xy, frame2_xy)

  return result
```

```
def get_keypoints_all_frames(json_list, first_frames):
  keypoints_all_frames = []
  for frame in range(first_frames):
      print('getting keypoints for frame: ', frame)
      with open(json_list[frame]) as fr:
        itemData = json.load(fr)
        peopleData = itemData['people']
        if len(peopleData) == 0:
          frameKeypoints = [0 for i in range(75)]
        else:
          frameKeypoints = peopleData[0]['pose_keypoints_2d']
      frame_xy = get_xy_coord_array(frameKeypoints)
      keypoints_all_frames.extend(frame_xy)
  return keypoints_all_frames
```

Neural Network Overview

Hyperparameters: 100 epochs, α=0.001

Final Model

- Three layers: input, hidden, and output
- Decided on a hidden layer size of **256 neurons**
- Used **ReLU** activation function for input and hidden layers
- Performs **Softmax** on the output to get the predictions in the range [0, 10]
- Error criterion is **Cross Entropy Loss**

Conclusions

Project Summary

Results

16-4-5 train-test-val split

Large values for mean abs. error

Predictions tended towards middle values (5-7)

Testing Results		Avg. MAE	1.27
Renegade	**All About Cake**	**Don't Start Now**	**Elastic**
test set mae: 1.11	test set mae: 2.18	test set mae: 1.40	test set mae: 0.39
predicted: 7.08	predicted: 7.90	predicted: 5.77	predicted: 6.98
actual: 6	actual: 9	actual: 3	actual: 7
predicted: 3.68	predicted: 6.08	predicted: 6.17	predicted: 7.03
actual: 4	actual: 5	actual: 7	actual: 7
predicted: 3.92	predicted: 5.37	predicted: 8.60	predicted: 7.13
actual: 2	actual: 1	actual: 8	actual: 6

Renegade					
Random seed:	0	123	222	345	678
	test set mae: 1.11	test set mae: 0.33	test set mae: 2.55	test set mae: 3.20	test set mae: 3.34
	predicted: 7.08	predicted: 5.00	predicted: 3.00	predicted: 4.63	predicted: 5.00
	actual: 6	actual: 5	actual: 5	actual: 2	actual: 2
	predicted: 3.68	predicted: 5.00	predicted: 6.88	predicted: 4.57	predicted: 5.01
	actual: 4	actual: 5	actual: 6	actual: 1	actual: 8
	predicted: 3.92	predicted: 5.00	predicted: 4.23	predicted: 4.60	predicted: 5.01
	actual: 2	actual: 4	actual: 9	actual: 8	actual: 1
Don't Start Now					
Random seed:	0	123	222	345	678
	test set mae: 1.33	test set mae: 0.34	test set mae: 1.08	test set mae: 1.33	test set mae: 1.40
	predicted: 5.01	predicted: 5.02	predicted: 4.59	predicted: 4.99	predicted: 5.77
	actual: 4	actual: 5	actual: 6	actual: 7	actual: 3
	predicted: 5.02	predicted: 5.02	predicted: 5.47	predicted: 4.99	predicted: 6.17
	actual: 7	actual: 6	actual: 7	actual: 3	actual: 7
	predicted: 5.01	predicted: 5.02	predicted: 5.29	predicted: 5.00	predicted: 8.60
	actual: 6	actual: 5	actual: 5	actual: 5	actual: 8
Elastic					
Random seed:	0	123	222	345	678
	test set mae: 1.49	test set mae: 1.16	test set mae: 2.61	test set mae: 0.39	test set mae: 1.67
	predicted: 6.49	predicted: 5.80	predicted: 6.74	predicted: 6.98	predicted: 5.00
	actual: 7	actual: 6	actual: 2	actual: 7	actual: 6
	predicted: 6.46	predicted: 5.87	predicted: 7.07	predicted: 7.03	predicted: 5.00
	actual: 3	actual: 8	actual: 8	actual: 7	actual: 7
	predicted: 6.50	predicted: 5.85	predicted: 5.86	predicted: 7.13	predicted: 5.00
	actual: 7	actual: 7	actual: 8	actual: 6	actual: 7
Savage					
Random seed:	0	123	222	345	678
	test set mae: 1.67	test set mae: 1.67	test set mae: 0.72	test set mae: 0.43	test set mae: 0.99
	predicted: 5.00	predicted: 5.00	predicted: 5.86	predicted: 5.75	predicted: 6.01
	actual: 4	actual: 6	actual: 7	actual: 7	actual: 6
	predicted: 5.00	predicted: 5.00	predicted: 5.87	predicted: 6.02	predicted: 5.99
	actual: 2	actual: 6	actual: 5	actual: 6	actual: 8
	predicted: 5.01	predicted: 5.00	predicted: 5.86	predicted: 5.99	predicted: 6.04
	actual: 6	actual: 2	actual: 6	actual: 6	actual: 7

Areas of Improvement

- Increase training/testing/validation data —> scrape data from TikTok instead of manual collection
- More data representing extreme ratings (1-3, 8-10)
- Consider other methods for rating videos
- Consideration for multiple dancers
- Dimensionality reduction of the feature space
- Investigating other models for multiclass classification (introducing other layers, experimenting with the activation function)