# Stance Classification using 3D Pose Estimation

Pierce Mayadag, Patrick Duane and Kai-Li Cheng

ARTICLE INFO

*Keywords*:
Pose Estimation
Stance Classification
Neural Networks

ABSTRACT

Pose classification has the potential to be a valuable tool in many fields, from security to sports. We explore using machine learning to classify martial arts poses in images. We analyze the performance of a variety of neural network configurations, including a specialized variation of convolution, as well as a number of statistical methods on the output of an established pose estimation algorithm. We found that a simple convolutional network consisting of a convolutional layer, a dense layer, and a softmax layer had the best performance of the neural network configurations, and that statistical methods were particularly bad at classification, with nearly all ANN configurations out performing the statistical methods.

## 1. Introduction

In the ever growing field of computer vision, we are always searching for ways for a computer to understand more of what it sees and to understand the world more completely and, perhaps, more humanly. One important part of social interaction is the ability to understand and interpret body language. Though the human brain easily detects and processes subtle nuances that we don't consciously recognize, computer systems are just beginning to figure out where the head is and where the feet are. This projects looks at taking a step beyond locating parts of the body and starts to take steps into the realm of interpretation. Being able to identify distinct poses from the nothing but an image would have a large range of applicability, from threat analysis for the government and security companies to correction tools for martial arts and sports, to postural analysis in physical therapy. In this report, we discuss our research, experimentation, and results as we explore how to get a computer to classify common, distinct martial arts poses.

## 2. Background

### 2.1. Problem Statement

Pose estimation is a developing technique in computer vision, that attempts to extrapolate the positions and orientation of a person and their limbs. Our project goal is to utilize an existing pose estimation technique to provide the input for a neural network that will classify poses in a 2-dimensional image as different martial arts poses.

### 2.2. Related Work

The exact problem worked on in this paper is unique, but the several others have created systems for action classification before. One method used 2D-RGB images to represent a 3D skeleton, with the 3D coordinates of joints mapped to RGB values, before applying a variety of classification techniques. They applied their techniques to datasets containing motion captured data, and classified common actions, including martial arts moves. With fine tuning, the highest

accuracy obtained was 92% on their best dataset, with 74% on their worst dataset. Both results were obtained using CNNs fed time series data Laraba, Brahimi, Tilmanne and Dutoit (2017).

Another work attempted to estimate 3D rotations and translations of a head using several 2D images within degrees of accuracy Janik, Harrison, Cailliet, Harrison, Normand and Perron (2007), but its goal is too different to compare. A closer work classified 6 yoga poses, using short videos. Kothari (2020) They also used a pose detection model for the original input, using the OpenPose library. This method achieved a best score of 99% classification accuracy. Kothari (2020)

## 3. Data Prepossessing

Each example in our data set is initially encoded as a 512x512 RGB image. Each image is then fed through an open source pose estimation model in order to extract the joint positions of the human subject in the photo. Nibali, He, Morgan and Prendergast (2018) For each example, the location of 17 joints is identified, and each joint position consists of coordinates in 3D space. The resulting 17x3 data encoding is referred to as a pose vector. The data set consisting of pose vectors is what our models take as input. (See figure 1)

By abstracting the 2D image into the 3D pose vector, we remove almost all data contained in the image, leaving behind only the most essential features. This abstraction should aid the network in classifying stances as well as drastically reduce the required computational power and memory. What we gain in abstraction is not free, however, as we will suffer from any inaccuracies in the pose estimation process.

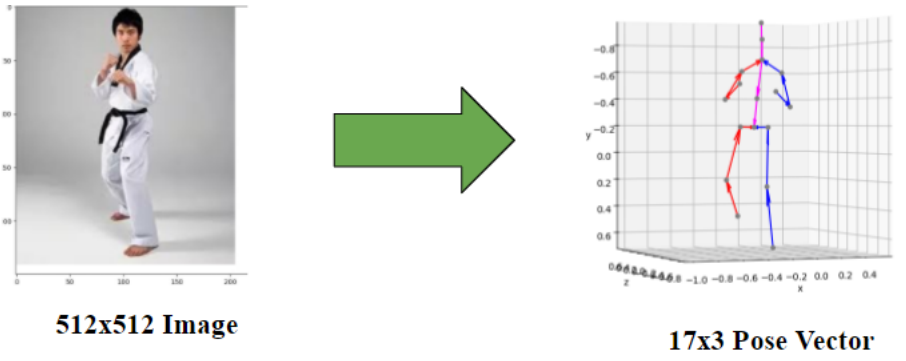


**Figure 1:** An example of one of the images from our dataset and its corresponding pose vector

For figure 1, it should be noted that the lines connecting the joints in the pose vector are purely for visual representation. The data that is given to our models has no information about how the joints are connected to one another. For this reason, we decided to create a modified version of our input encoding that would encapsulate this information. For each pair of connected joints in the pose vector, we added an additional 10 evenly spaced data points along the line

connecting the joints. Given that there are 16 joint connections in the pose vector, this brought our data encoding up to a size of 177x3. (See figure 2)

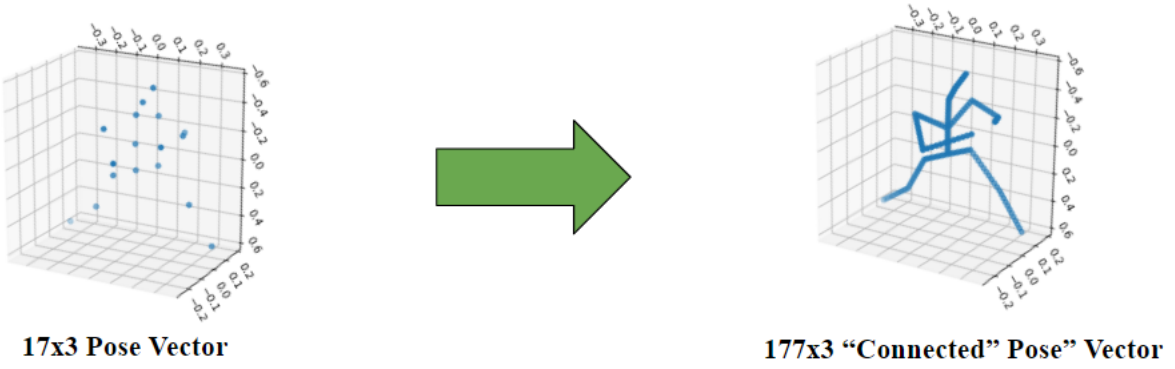


**Figure 2:** Comparison of our original pose vector encoding to our connected version

## 4. Methodology

### 4.1. ANN

We used several variations of the back-propagation ANN to classify the pose-vectors in our data set. Each of the following models were implemented in Tensorflow, using "softmax_cross_entropy" as our loss function and "adam" as our optimizer. Models were trained using a batch size of 2 for 70 epochs. The weights in the networks were initialized randomly, so each model was trained from scratch 5 times to calculate an average testing accuracy. 80% of our dataset was used as training data and the remaining 20% was used as testing data. (See figure 3)

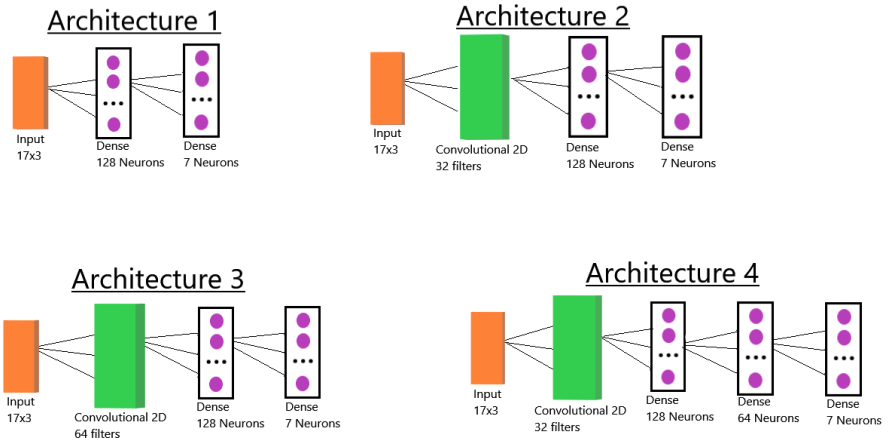


**Figure 3:** 4 ANN architectures of varying complexity

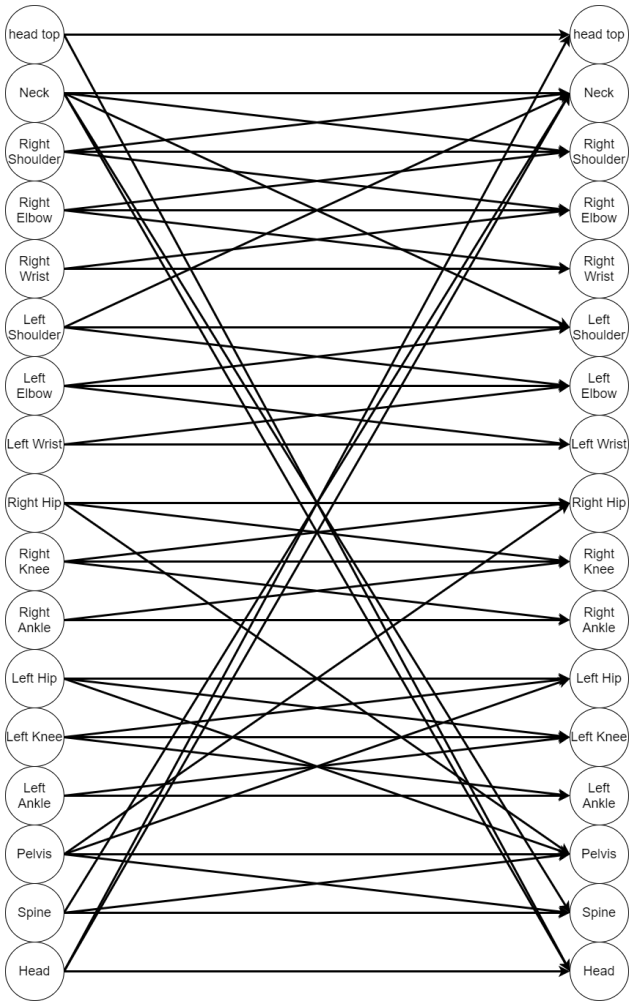


**Figure 4:** *Prope Iuncturam* Layer Visualization

Of the 4 architectures, Architecture 2 performed the best. We later tested Architecture 2 using our connected pose vector data set. Despite this data set theoretically containing additional useful data points, the model scored slightly lower than it did with the original data set.

## 4.2. PI Layer

The *Prope Iuncturam* (PI) layer is a custom layer that aims to specialize the concept of convolution for our problem. Just as an image tends to have connections between nearby pixels, we suspect our structure would tend to have connections between nearby joints. This layer of the ANN would have the same number of outputs as it does inputs, and each node in the layer would have a connection from all adjacent inputs, including itself (for example, the right shoulder would be connected to the neck, itself, and the right elbow).

### 4.3. Image Classifier

A CNN was created as a control to show the effect of the 3D pose estimation step. It bypassed the transfer learning (pose estimation) prepossessing, and attempts to classify the stance merely based on an image. The network is composed of 3 convolutional layers with max pool layers in between (see figure 5). The network ends in a dense relu layer connected to the output classification layer. Attempts to avoid over-fitting included adding a droppout layer after the convolutional layers, to randomly dropout output units to prevent over-fitting.

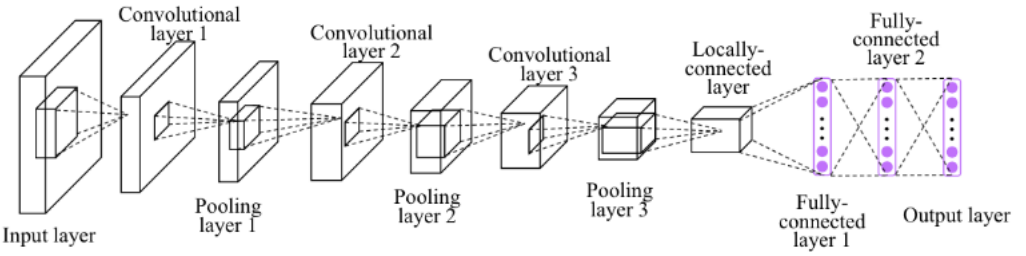


**Figure 5:** The architecture of the Image Classification CNN. Three Convolutional layers, separated by max-pooling - Image Credit Damien (2018)

### 4.4. Statistical Methods

Several statistical methods (non ANN) were applied to the problem to compare performance with the ANNs. These were implemented using packages from scikit-learn, with the exception of XGBoost which has its own package. Where the ANNs above could process the data in its original form (17x3) the following models received that data as (51xSamples) with a feature corresponding to each dimension (x,y,z) of each limb.

#### *4.4.1. KNN*

*K-Nearest-Neighbor* classifies instances by mapping each training instance, then classifying the new instance as belonging to the class of it's kth nearest neighbor. KNN had the simplest implementation and fastest runtime. Even though the data is spatial, the hidden relations between two different stances are fairly subtle and KNN may struggle here. James, Witten, Hastie and Tibshirani (2013)

#### *4.4.2. PCA*

*Principal Component Analysis* reduces dimensions by combining and extracting important features. James et al. (2013) PCA was applied before KNN, as KNN is susceptible to the curse of dimensionality. Reducing dimensions never significantly increased classification accuracy, but also didn't reduce classification accuracy until 26 or more dimensions were removed (see fig 6). This confirms the rule of thumb that 5 instances per dimension are required to escape the effects of the "curse", and we train on 617 instances(12 instances per dimension). PCA also alows us to visualize data points from large dimensions. See figure 10.

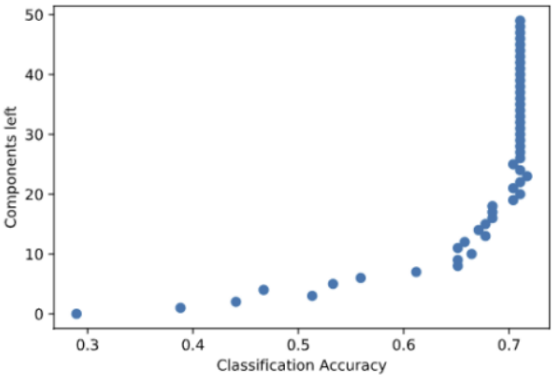


**Figure 6:** Components left in the model vs Classification Acc. No significant increase in Classification Accuracy, which starts decreasing after 24 components are removed

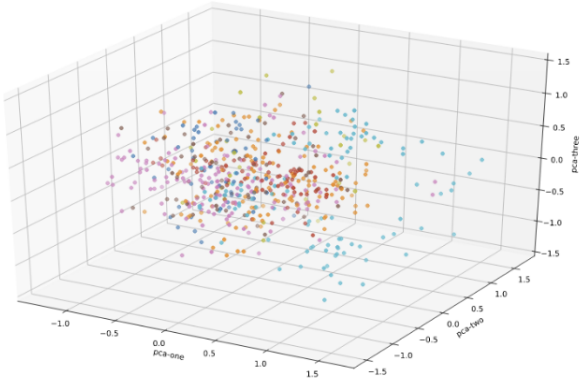


**Figure 7:** The 3 most significant principal components used to visualize our data. Each color is a different stance

### *4.4.3. SVM*

*Support Vector Machines* attempt to define the ideal hyperplane to separate target features. SVMs maximize decision boundaries, to create a large margin between target features (with the least number of samples inside the boundary). SVM was implemented using the RBF (Radial Basis Function) kernel. Kernel SVMs are often preferred to linear functions, since they can handle nonlinear decision boundaries. Looking at fig 10, it appears difficult to separate different classes with a linear decision boundary. James et al. (2013)

### *4.4.4. CART - Classification and Regression Trees*

*Decision Trees* have branches corresponding to feature values and end in terminal(leaf) nodes corresponding to target values. There are several methods of determining split points, but for this project, splits in the tree are determined using Gini impurity. The Gini Impurity (GI) gives how likely an instance will be missclassified, if randomly classified according the the distribution. The split is one that that maximizes the Information Gain (IG) (difference between GI

of the parent and the sum of the GI's of the children. Strobl, Boulesteix and Augustin (2007)

$$GI(K) = \Sigma_{i \in C} P_{i,K}(1 - P_{i,k}) \qquad IG(K) = GI(K_{parent}) - \Sigma_{j=i}^{m} \frac{N_j}{N} GI(K_j)$$

K = a class, C = all classes, N = number of samples, $N_j$ = number of samples at the jth node, $P_{i,k}$ = probability of class K having class i.

#### *4.4.5. Random Forest*

Random Forest is a form of ensemble learning, where the results of multiple decision trees are created using bootstrapping (bagging). Bagging sub-samples the data with replacement to create more trees. RF improves over regular bagged trees by sampling a set of m random predictors each split. This keeps the trees less correlated to increase generality. An instance is passed to each tree, and the results from each tree in the forest are combined using majority voting. James et al. (2013)

#### *4.4.6. XGboost*

Gradient Boosting Machines are also examples of decision tree ensemble learning. The GBM begins with a single tree, then calculates some loss function, then adds additional trees to minimize loss. The results from each tree are combined using the weighted aggregate of the predictions made by each model. XGboost is a GBM algorithm that uses L1 & L2 regularization to improve generalization. It also prunes trees and implements parallel processing. Glandir (2019)

## 5. Results

### 5.1. Performance Metrics

In multi-class classification, each of the following metrics will be calculated for a single class, then combined as a weighted average (weighted by the number of instances in that class). The Accuracy, Specificity, Negative Predictive Value (NPV), False Positive Rate (FPR), Precision, and Recall metrics are considered, see equations 5.1. Figure 8 shows a confusion matrix for a multi class problem, and how the TN, FN, FP, and TP values are extracted. This figure also exposes a problem with the Accuracy, Specificity, NPV and FPR metrics. They relay on TN values, which can inflate their score. A misclassified instance will still increase the metric, since even though the label wasn't correctly predicted for its class, it counts as a TN for every other class. The Prediction and Recall metrics are much more important predictor in a multi-class classification problem than the others. F1 Score is a harmonic mean of precision and recall, but shouldn't be taken as an absolute in every case (classifying a sick person as healthy has a different

weight that classifying a healthy person as sick).

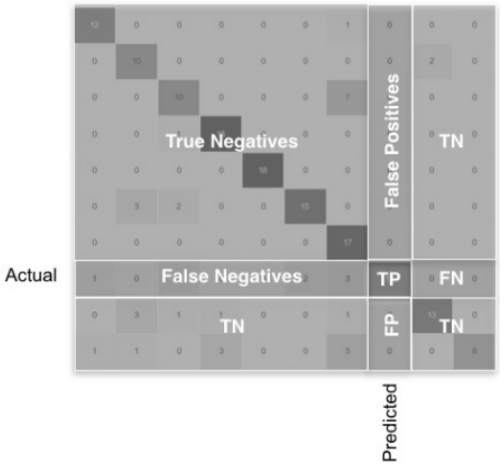


**Figure 8:** Multi Class Confusion Matrix

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \qquad Specificity = \frac{TN}{TN + FP}$$

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FP}$$

$$NPV = \frac{TN}{TN + FN} \qquad FPR = \frac{FP}{FP + TN}$$

## 5.2. Comparison and Evaluation

Focusing on the precision and recall metrics, the best performance come from the CNN that only took in our joint positions, with a precision of 0.8158 and a recall of 0.8190. The KNN, PCA, SVM methods showed similar performance, with 0.73 precision. The connected CNN (same CNN with connected dataset) performed slightly worse than our standard CNN, where we expected some improvement. Fine tuning the CNN to the dataset directly could help resolve these issues. Likewise the PI layer performed better than the statistical methods (except XGBoost). XGBoost performed remarkably well, training and executing faster than the CNNs and maintaining similar performance ( 0.75 Precision). Notably, the Image ANN performed the worse on every metric, with a dismal 0.5 precision. This number still may actually be inflated, since photos of similar stances had similar backgrounds, due to limitations in data gathering. Where this background issue wouldn't effect our joint position data (all background and color is removed in the transfer learning step), it would effect how the image classifier perceived the stances.

The related work examined in section 2.2 is only a vaguely similar problem, classifying a sequence of motions from a 3D motion captured data. However, we did achieve similar performance to their 70% - 90% range. We performed worse than the yoga pose classifier, which may show the power of time series data and LSTM.

| | Accuracy | Recall | Specificity | Precision | NPV | FPR |
|---|---|---|---|---|---|---|
| KNN | 0.9058 | 0.7369 | 0.942 | 0.7178 | 0.9416 | 0.058 |
| PCA | 0.9069 | 0.7397 | 0.9427 | 0.7216 | 0.9421 | 0.0572 |
| SVM | 0.896 | 0.7326 | 0.9298 | 0.7088 | 0.9379 | 0.0702 |
| IMG ANN | 0.789 | 0.5182 | 0.9215 | 0.5 | 0.8701 | 0.15 |
| RF | 0.8901 | 0.7282 | 0.9637 | 0.6982 | 0.9353 | 0.0784 |
| ANN | 0.9405 | 0.819 | 0.9521 | 0.8158 | 0.9631 | 0.0363 |
| Connected ANN | 0.9294 | 0.7986 | 0.9537 | 0.7763 | 0.9591 | 0.0463 |
| XGBoost | 0.9148 | 0.7749 | 0.9428 | 0.7568 | 0.9487 | 0.0572 |
| PI ANN | 0.9069 | 0.7338 | 0.94071 | 0.7197 | 0.944 | 0.0592 |
| DT | 0.8511 | 0.5983 | 0.9021 | 0.5734 | 0.9096 | 0.0593 |

**Figure 9:** Different Performance metrics compared to each model. The Recall and Precision metrics are more indicative of actual performance

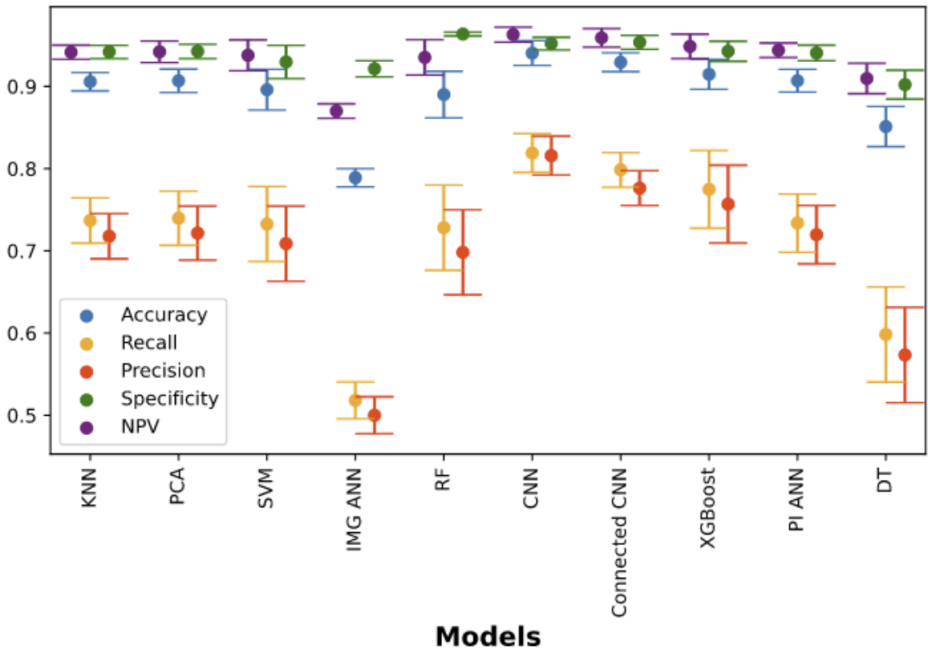


**Figure 10:** The 3 most significant principal components used to visualize our data. Each color is a different stance

### *5.2.1. PI*

The PI layer connected to a dense layer with 128 nodes followed by a softmax layer seemed to have the best performance out of all models we tried. This is the model whose performance metrics were reported above. We tried

a few other models in hopes of coaxing a little better performance out of the layer. We tried a model following the pattern of the image classification model, but each added layer only decreased the effectiveness. (Each model ended the same: a 128 node dense layer and a softmax layer). We also tried stringing two of these layers together, but this also decreased effectiveness.

## 6. Conclusion

Identifying the stance someone takes has a variety of applications, from training tools and video games, to postural analysis in physical therapy. We developed a cheap method for classifying various martial arts stances from a single photo. Where similar work required multiple cameras, or time series data, we leveraged the power of transfer learning to gain 3D spatial data from a 2D image. We used this data to great effect, correctly classifying stances up to 30% more accurately using Convolutional Neural Networks than a standard image classifier. This project demonstrated that accurately classifying stances using cheap methods is within our grasp.

## 7. Future Work

One aspect of our project that needed more testing was with the connected pose vector data set. This variation to our original data set would in theory add useful information that was not previously represented. Unfortunately, due to time constraints, we were only able to test this dataset with our Artificial Neural Network. Although the modified data set performed slightly worse than the original, we believe that more testing using other classification models may have produced better results.

The largest issue with this method comes from it's relatively small dataset. Even flipping images to double the size of our dataset, we only had 600 samples. Additional improvements to accuracy could come from using time series data, looking at a live video instead of an image will give more context to each stance. This would allow us to test more labels and expand the generality of our models.

Future work also includes developing applications to take advantage of this intelligence.

## CRediT authorship contribution statement

**Pierce Mayadag:** Conceptualization, Methodology, Software, Formal Analysis, Investigation, Writing - Original Draft & Review & Editing. **Patrick Duane:** Conceptualization, Methodology, Software, Formal Analysis, Investigation, Data Curation, Writing - Original Draft & Review & Editing. **Kai-Li Cheng:** Conceptualization, Methodology, Investigation, Software, Validation, Formal analysis, Investigation, Data Curation, Writing - Original Draft, Project Administration.

## References

Damien, A., 2018. Convolutional neural network (tf.layers/estimator api). URL: https://wizardforcel.gitbooks.io/tensorflow-examples-aymericdamien/content/3.05_convolutional_network.html.

Glandir, S., 2019. Gradient boosting and xgboost. URL: https://opendatascience.com/gradient-boosting-and-xgboost/.

James, G., Witten, D., Hastie, T., Tibshirani, R., 2013. An introduction to statistical learning. volume 112. Springer.

Janik, T.J., Harrison, D.E., Cailliet, R., Harrison, D.D., Normand, M.C., Perron, D.L., 2007. Validity of a computer postural analysis to estimate 3-dimensional rotations and translations of the head from three 2-dimensional digital images. Journal of manipulative and physiological therapeutics 30, 124–129.

Kothari, S., 2020. Yoga pose classification using deep learning .

Laraba, S., Brahimi, M., Tilmanne, J., Dutoit, T., 2017. 3d skeleton-based action recognition by representing motion capture sequences as 2d-rgb images. Computer Animation and Virtual Worlds 28, e1782.

Nibali, A., He, Z., Morgan, S., Prendergast, L., 2018. 3d human pose estimation with 2d marginal heatmaps. arXiv preprint arXiv:1806.01484 .

Strobl, C., Boulesteix, A.L., Augustin, T., 2007. Unbiased split selection for classification trees based on the gini index. Computational Statistics & Data Analysis 52, 483–501.