Project Report - Sport Activity Pose Classification

Project members: Jacquelyn Nehra, Ella Frohock, Cameron Otten, Tommy Whaley, Eric Wen

Introduction and Motivation

The purpose of this project was to use Machine Learning to perform pose estimation. Given some input describing a human subject, the model would be able to correctly classify what pose the subject was in. The motivation was to gain practical experience in creating a machine learning model and explore image classification.

Problem definition

The input consisted of 480x270 .JPG images of subjects engaged in various activities. These images were annotated with the subject's joint locations, each joint being labeled with the x,y position in the image and boolean to indicate whether the particular joint was visible. The annotations labeled the head and the left and right shoulders, elbows, wrists, hips, knees, and ankles. The boolean accounts for cases where a joint was obscured (as in the image below), out of frame, or otherwise invisible. The annotations also included the class label.

Example of Activity Image

The output of the model was multiclass, and would determine one of fifteen actions to be most likely in the image. Inputting the data corresponding to the above image should output Baseball Pitch, for example. Other classes include Tennis Serve, Bowl, Guitar Strum, and more.

Solution and Data

Machine learning was determined to be a desirable way to solve the multiclass problem due to the complexity and inexact nature of a classification function as well as the variation and complexity of the input. Machine learning has been successfully used to solve problems of this nature before. Given that we had extensive labeled data, supervised machine learning algorithms were going to best fit our needs.

The input data was collected from the Penn Action Dataset, a dataset composed by the University of Pennsylvania [1]. The dataset contained 2326 pre-annotated video sequences of people doing the various activities, and was available for download.

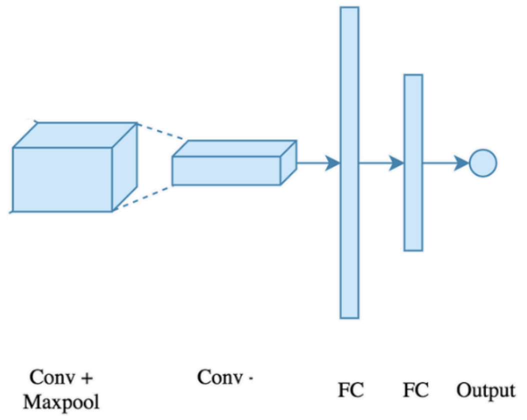
In terms of the usage of the data in machine learning models, each input has a series of .jpg files and a series of .mat files corresponding to the image for a frame and the annotations for that frame, respectively. An image is chosen from near the middle of the sequence to represent the action. The image is resized to 224x224 and represented as a tensor with normalized values. The annotations were also represented by tensors.

Chosen Algorithm Overviews

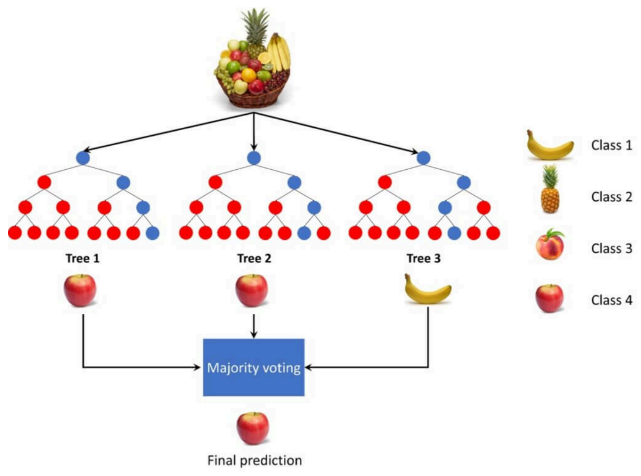
Two working models were developed. The first was a Convolutional Neural Network (CNN) which took only the image data as an input. The first convolutional layer took the three input channels to 16 output channels, applied ReLU and then max pooling. The second layer brought it to 32 output channels and again applied ReLU and max pooling. These layers extract image features such as edges. The tensor is then flattened and put through a linear layer of size 128 to learn patterns in the extracted features. One more ReLU is done and then a final layer of 15, one to match each output, is added. These final outputs are then converted into probabilities using a log softmax function, the highest probability being what the image is classified as.

The second model developed was a Random Forest Classifier. Random Forest is a supervised machine learning algorithm that builds multiple decision trees during training and combines their outputs (either via majority voting for classification or averaging for regression) to make predictions. Our task was classification, so our model uses majority voting. It is robust to overfitting, handles high-dimensional data well, and performs effectively even with complex data. For our sport activity pose classification problem, Random Forest can process the engineered features derived from joint annotations to classify given activity poses into different classes.

Example CNN Layers [2]



Example Random Forest Process [4]



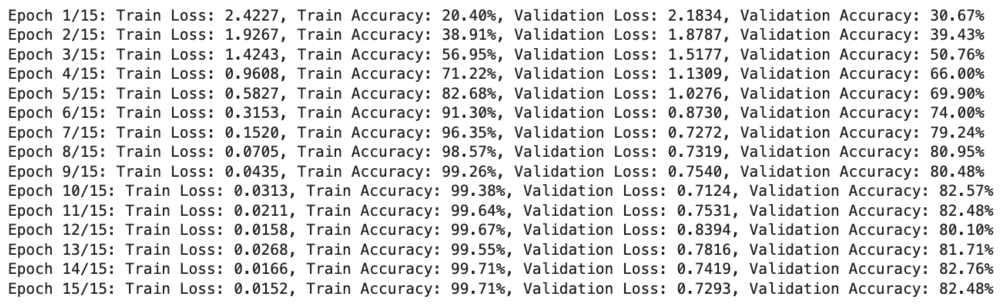
Implementation and Techniques

For the Random Forest model, we utilized Scikit-learn, a core library for machine learning in Python. With this library we initialized and ran the classifier and were able to use functions from sklearn.metrics to evaluate model performance and adjust hyper parameters.

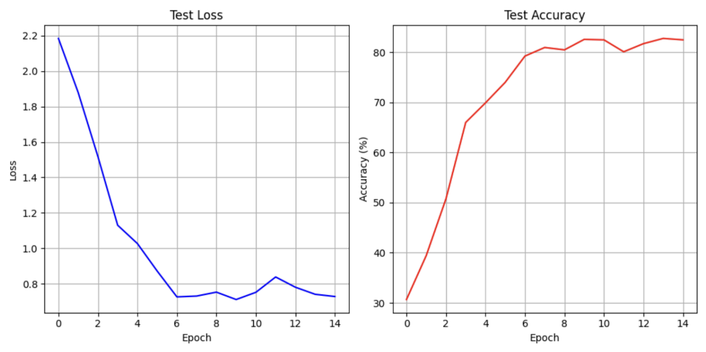
For CNN, we used PyTorch to transform the data as well as create the CNN module. Pillow was used to open the images. Finally, Scikit-learn was used to split the data into train and test sets. The CNN module consists of two convolutional layers and two linear layers. The small number of layers helps to preserve the input signal.

Results

The CNN resulted in an accuracy of about 82% after 15 epochs. Both the training and testing loss and accuracy are shown in the figure below. Additionally, there is a plot of both the test loss and accuracy.



```
Epoch 1/15: Train Loss: 2.4227, Train Accuracy: 20.40%, Validation Loss: 2.1834, Validation Accuracy: 30.67%
Epoch 2/15: Train Loss: 1.9267, Train Accuracy: 38.91%, Validation Loss: 1.8787, Validation Accuracy: 39.43%
Epoch 3/15: Train Loss: 1.4243, Train Accuracy: 56.95%, Validation Loss: 1.5177, Validation Accuracy: 50.76%
Epoch 4/15: Train Loss: 0.9608, Train Accuracy: 71.22%, Validation Loss: 1.1309, Validation Accuracy: 66.00%
Epoch 5/15: Train Loss: 0.5827, Train Accuracy: 82.68%, Validation Loss: 1.0276, Validation Accuracy: 69.90%
Epoch 6/15: Train Loss: 0.3153, Train Accuracy: 91.30%, Validation Loss: 0.8730, Validation Accuracy: 74.00%
Epoch 7/15: Train Loss: 0.1520, Train Accuracy: 96.35%, Validation Loss: 0.7272, Validation Accuracy: 79.24%
Epoch 8/15: Train Loss: 0.0705, Train Accuracy: 98.57%, Validation Loss: 0.7319, Validation Accuracy: 80.95%
Epoch 9/15: Train Loss: 0.0435, Train Accuracy: 99.26%, Validation Loss: 0.7540, Validation Accuracy: 80.48%
Epoch 10/15: Train Loss: 0.0313, Train Accuracy: 99.38%, Validation Loss: 0.7124, Validation Accuracy: 82.57%
Epoch 11/15: Train Loss: 0.0211, Train Accuracy: 99.64%, Validation Loss: 0.7531, Validation Accuracy: 82.48%
Epoch 12/15: Train Loss: 0.0158, Train Accuracy: 99.67%, Validation Loss: 0.8394, Validation Accuracy: 80.10%
Epoch 13/15: Train Loss: 0.0268, Train Accuracy: 99.55%, Validation Loss: 0.7816, Validation Accuracy: 81.71%
Epoch 14/15: Train Loss: 0.0166, Train Accuracy: 99.71%, Validation Loss: 0.7419, Validation Accuracy: 82.76%
Epoch 15/15: Train Loss: 0.0152, Train Accuracy: 99.71%, Validation Loss: 0.7293, Validation Accuracy: 82.48%
```



The Random Forest model reached an accuracy of ~83%. After training several versions of the model on different numbers of estimators, the amount of estimators that maximized accuracy was n=1350. A feature importance analysis revealed that none of the joint features had particularly high importance in the model, and removal of below average importance features resulted in a loss of ~3% accuracy. A randomized search CV was run to optimize hyper parameters, but it also failed to improve on the model accuracy. Ultimately, the model with optimized n estimators reported the best accuracy. 9 of the 15 classes had F1 scores over 0.8.

Related work

One of the highest accuracy models that already exists for this problem uses a combination of ResNet, a custom module called WASP, a decoder layer, and some final convolutional layers [4]. This model extrapolated joint connections and bounding boxes as well as labeling the image. This model uses pre-processing of the joint annotations to create heatmaps. These heatmaps are then used to train the decoder to generate its own heatmaps. This allows it to reach a 98.4% accuracy.

Conclusion

The models were fairly successful in classifying the images, with both achieving accuracies of ~82-83%. It was unfortunately not to the degree desired, as the goal had been to reach a 90% accuracy. Nonetheless, the goal of gaining practical experience in building and tuning large machine learning models was a success. There are some opportunities for improvement, especially in regard to the combined model which trains on the image and the annotations, which we hope would have been able to improve upon the success seen in our other models. Considering the 95%+ accuracy that state of the art models can achieve, there are adjustments that can be made, such as training with a larger amount of data, additional preprocessing, or the use of LSTMs to improve results in the future.

References

[1] Weiyu Zhang, Menglong Zhu and Konstantinos Derpanis, "From Actemes to Action: A Strongly-supervised Representation for Detailed Action Understanding" *International Conference on Computer Vision* (ICCV). Dec 2013.

[2] A. Dertat, “Applied Deep Learning - Part 4: Convolutional Neural Networks,” *Medium*, Nov. 8, 2017. https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

[3] S. Zivkovic, “#012 Machine Learning - Introduction to Random Forest - Master Data Science 30.08.2022,” *Master Data Science*, Aug. 30, 2022. https://datahacker.rs/012-machine-learning-introduction-to-random-forest/

[4] B. Artacho and A. Savakis, “UniPose: Unified Human Pose Estimation in single images and videos,” *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7033–7042, Jun. 2020. doi:10.1109/cvpr42600.2020.00706