Human Body Marker Augmentation

(Code Repository: https://github.com/cterrill26/augmenter-cs230)

Akshit Goel
SUID:006599410
Electrical Engineering
akshit@stanford.edu

Rhythm D Patel
SUID:006439126
Civil and Environmental Engineering
rhythm@stanford.edu

Caleb Terrill
SUID:006311354
Electrical Engineering
cterrill@stanford.edu

1. Abstract

Quantitative analysis of human motion can offer comprehensive information to approach applications in the domain of biomechanics such as sports performance and physical rehabilitation. In this project, we propose the use of a modified natural language model. GPT-2. to augment the markers generated by the OpenPose model and generate a more comprehensive set of body markers. These markers have been defined through previous research as necessary for understanding human motion dynamics. In addition to OpenPose markers, we also propose to use body measurements as additional input features to the model for more accurate prediction of augmented markers.

2. Introduction

Evaluating human motion dynamics is important to understand and manage musculoskeletal and neuromuscular diseases. Predicting the body pose from an input video or image is important to quantitatively analyze the dynamics of body motion. However, this quantification requires knowledge about the position of a set of specific human body markers and their movement during motion. The existing pose estimation solutions like OpenPose give a small set of markers which is not sufficient to guide biomechanical research. To overcome this challenge, deep learning is being used to augment the markers to generate a more comprehensive set of body markers. Uhlrich et al. (2022) [1] have used recurrent neural architectures (long short-term memory networks - LSTM) for this purpose..

Transformer architectures (Ashish Vaswani et al. 2017) [2] have traditionally been used for sequence problems such as modeling natural language (Brown et al., 2020) [3], proteins (Jumper et al., 2021) [4], and many other applications. However, in recent research Transformer models have been employed to solve video sequence problems such as object detection (Carion, Nicolas, et al. 2020) [5] and 3D human motion synthesis (Petrovich et al., 2021) [6]. Lu et al. (2021) [7] showed that transformer models like GPT-2 (Radford et al., 2019) [8] which have been pre-trained on natural language data, can also achieve good results for a range of tasks like protein fold prediction and image classification. Taking inspiration from this, in this project we adapted the pretrained natural language model GPT-2 for this novel application of human body marker augmentation.

3. Related Work

Wu Tsai Human Performance Alliance at Stanford University has developed OpenCap, a software package leveraging computer vision, deep learning, and musculoskeletal modeling and simulation to quantify human movement from smartphone videos. As part of the OpenCap pipeline, the researchers have trained recurrent neural networks (long short-term memory networks - LSTM) to predict the position (3D coordinates) of a comprehensive set of anatomical markers on the human body from a reduced set of markers identified from videos using pose estimation models (eg, OpenPose) [1]. The training and validation RMSE are 0.0054 and 0.0063 respectively. The anatomical markers predicted from the current LSTM require more accuracy between different joint angles. Also, the model seems to behave poorly for human moments which it has not been trained on or are captured from different camera angles. As proposed by Lu et al. (2021) [7], we adapted the Frozen Pretrained Transformers (FTP) which is a pre-trained language model GPT-2 (Radford et al., 2019) [8] as our baseline for the project. Our goal is to compare the LSTM model's performance (Uhlrich et al. 2022) [1] to transformers (GPT-2) on predicting anatomical markers.

4. Dataset and Data Visualization

To train the network, we used a dataset consisting of motion capture data processed with OpenSim. The data was extracted from open-source repositories where each motion file came with an OpenSim model. For each OpenSim model virtual markers corresponding to the video keypoints are added on the one hand and the anatomical markers on the other hand. For each motion file, the 3D trajectory of virtual markers are extracted and these trajectories are used as features and labels to train networks. For each marker, these trajectories are sequenced at 0.5 s (30 frames at 60 Hz). We trained our network on a total of 121555 sequences (17 hrs of data). There is also a scope to train the model for 9 more datasets (total of 972 hrs) but due to paucity of time, it could not be done for this project..

Input and Output vectors: As shown in the figure below, virtual keypoints are our input feature markers for the model. Each input example in our baseline model consists of (x,y,z) coordinates of 15 virtual keypoints. There are 2 additional features representing the height and weight of the subject. Since each marker is sampled for 30 frames, the input is a matrix of dimension (30 x 15*3 +2) i.e (30 x 47). From these 15 markers we wish to predict the position of 35 anatomical markers. So the output dimension is (30 x 35*3) i.e. (30 x 105). As a novel addition to the project, as we describe in section 5.2 we also added 16 additional markers in the input vector, so the input dimension becomes (30 x 63) for that part of the project.

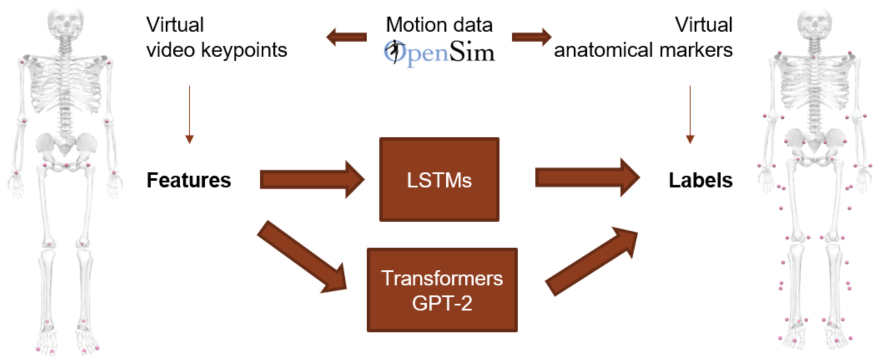

Fig. 1 Input Features and Augmented Features

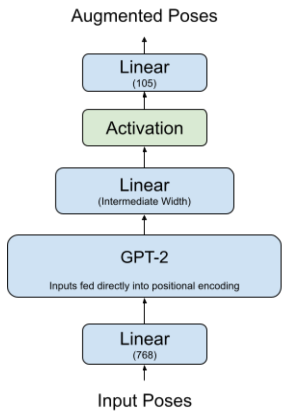

Fig. 2 Model Block Diagram

5. Methods

We propose two novel ideas in this project to improve the performance achieved by OpenCap. The first is using a more powerful model at the core of our learning algorithm—a transformer instead of an LSTM, and the second is adding more features as inputs to our model to ease the learning process. These techniques are described in the next two subsections.

5.1 Model Overview

At the core of our learning algorithm is a GPT-2 transformer. The GPT-2 transformers are a series of powerful models trained and released by OpenAI for English text prediction. There are multiple different versions of GPT-2 and trained parameters available online. The one we chose to use is the smallest one available on Hugging Face with 12 layers, 768 hidden units, 12 attention heads, and 117 million total parameters [9]. Specifics about the GPT-2 models can be found in the GPT-2 paper released by OpenAI [8]. As shown by Lu et al. [7], pretrained transformers are able to generalize to use cases other than natural language processing without fine tuning the self-attention and feedforward layers. Thus, we expect GPT-2 to be useful for our task of pose augmentation.

Since the input to our model is a series of 3D coordinates, we do not need to use a tokenizer like in most use cases of GPT-2 to convert words to vectors. Instead, we simply feed the input positions through a linear layer with 768 outputs, and this is fed directly into GPT-2 at the stage prior to positional encoding. The output of the GPT-2 model is then fed through a linear layer followed by an activation function, and a final linear layer produces the resulting augmented poses. When training, all the layers of GPT-2 are frozen except for the parameters in the position encoding and layer normalization layers. A block diagram of our model is shown in Fig. 2.

5.2 Additional Input Features

One problem that the OpenCap group faced when attempting to augment poses was that the input 3D joint locations were noisy. Joint locations moved around from frame to frame despite little movement in the subject's position being represented. In order to take this into account during training and make their model more resilient to noise, the OpenCap researchers added normally distributed noise to the input and output pose data during both training and evaluation. In addition to the input pose, the OpenCap researchers also fed subject height and weight into their models. These values were the same for all frames in an input sequence, and thus were noise free.

We decided to take the idea of additional body measurements such as height and weight one step further by adding many more of them in our experimentation. We wondered if with more noise free inputs, whether or not our model would be able to take this additional information about the subject into consideration and dampen the effect of the positional noise. The additional metrics we used were body measurements such as shoulder width or hip width, and they could be reported by a user of the augmentation application or estimated in a calibration flow while the subject remains still. The additional measurements we chose are listed in Table I below.

Join 1	Neck	Neck	RShoulder	RShoulder	LShoulder	LHip	LHip	RHip
Join 2	RShoulder	LShoulder	LShoulder	RHip	LHip	RHip	LKnee	RKnee

Join 1	RKnee	LKnee	RAnkle	LAnkle	RHeel	LHeel	RHeel	LHeel
Join 2	RAnkle	LAnkle	RHeel	LHeel	RSmallToe	LSmallToe	RBigToe	LBigToe

Table I:. Endpoints of Additional Input Features

Note that no points are used from the arms, as they are not part of the input and output set of points that we used. Since we did not have access to the exact measurements of the subjects in our dataset, we were forced to estimate these additional features from the input dataset.

5.3 Baseline:

Our baseline implementation consists of the architecture described in 5.1. According to Lu et al. (2021) [7] a pre-trained transformer language model can obtain strong performance on a variety of non-language tasks without having to finetune the self-attention and feedforward layers. So for our baseline implementation we relied on a pre-trained GPT-2 model and trained our input and output layers. The baseline choice of hyperparameters is shown in the hyperparameter exploration in section 6.1.

In this project, we have targeted to achieve a low RMSE between our predictions and the actual coordinates. The performance of our baseline model is as following:

- Training Set: RMSE: 0.0113
- Validation Set: RMSE: 0.0074

Based on our experiments, we noticed that the presence of the dropout layer is responsible for getting a bigger training loss as compared to validation loss.

After implementing a baseline model, we performed multiple experiments and tuned the hyperparameters to improve the performance of our model. We then used these tuned hyperparameter values to evaluate the performance of our model with and without the addition of the new input features and with either a frozen or unfrozen GPT-2 model.

6. Experiments and Results

6.1 Experiments

To improve the performance of the baseline model, we identified 6 hyperparameters that we believed would help us improve our RMSE. For each hyperparameter, where possible we identified values above and below the baseline value. For hyperparameters like the number of nodes in the second-last linear layer, we decided to choose our values in the range (105, 768) to provide an intermediate step for mapping the (30 x 768) GPT-2 output to a (30 x 105) prediction. For each hyperparameter, to make clear comparisons we kept all other hyperparameters at the baseline value. The baseline value, experimentation values, and final optimal value for each of the hyperparameters are summarized in the table below. In this table the FC layer refers to the second last output layer.

S.No.	Hyperparameter	Experiment values	Baseline Value	Optimal Value
1	FC Layer - Activation	Sigmoid, Relu, Tanh	Relu	Sigmoid
2	FC Layer - No. of Hidden Units	128, 256, 512	256	512
3	Batch Size	64, 128, 256	128	256
4	Loss Function	L1, RMSE, Log cosh	RMSE	RMSE
5	Learning Rate	0.01, 0.001, 0.0001, 0.00003	0.001	0.00003
6	FC Layer - Dropout Probability	0.1, 0	0.1	0

Table II: Hyperparameters - Experiment and Baseline values

The choice of these hyperparameters was dictated by the fact that we decided to use a pretrained language transformer model as given in Lu et al. [7]. Tuning the hyperparameters on unfrozen GPT-2 layers was not feasible for this project considering its long runtime and the relatively short duration of the project. The hyperparameters that we tuned were related to our fully connected layers. We later used these tuned hyperparameters to train our model by unfreezing the GPT-2 model to give us improved performance as we hoped it would give the model more scope to learn from the input features. Using these tuned hyperparameters, we also experimented and evaluated the impact of incorporating new additional input features to our model. The results for these experiments are summarized in the next section.

6.2 Results

The following table shows the results of the tuning experiments we ran to find the optimal values of our 6 hyperparameters.

Parameter	Value	Train RMSE Loss	Eval RMSE Loss
All Parameters	Baseline	0.01132028	0.0073944
FC Layer - No. of Hidden Units	128	0.013095	0.00756193
FC Layer - No. of Hidden Units	512	0.01018912	0.00718031
FC Layer - Activation	Sigmoid	0.00948898	0.00653899
FC Layer - Activation	Tanh	0.00951608	0.0090287
Loss Function	L1	0.01281141	0.00751779
Loss Function	Log Cosh	0.01139723	0.00764623
Learning Rate	0.01	0.03810914	0.04857676
Learning Rate	0.0001	0.01168843	0.00706145
Learning Rate	0.00003	0.0108818	0.00650075
Batch Size	64	0.0113273	0.00734065
Batch Size	256	0.01142311	0.00702028
FC Layer - Dropout Probability	0.0	0.00669697	0.00707537

Table III: Hyperparameter Tuning Results

From these results, we chose the optimal values for each hyperparameter shown in the rightmost column of Table II. Using the tuned hyperparameters, we launched four final rounds of training and evaluation with and without our novel inputs and with GPT-2 frozen or not frozen. The results of these experiments are listed in Table IV, and the corresponding RMSE loss vs epoch graphs are shown in Fig. 3 alongside the performance of the baseline model. We refer to these runs with the labels a) through d) from this point forward.

Label	Novel Inputs Included	GPT Frozen	Train RMSE Loss	Eval RMSE Loss
a)	No	Yes	0.00621911	0.00602557
b)	No	No	0.00410163	0.00673851
c)	Yes	Yes	0.00646864	0.00625424
d)	Yes	No	0.00484653	0.00635123

Table IV: Final Training And Evaluation Results With Tuned Hyperparameters

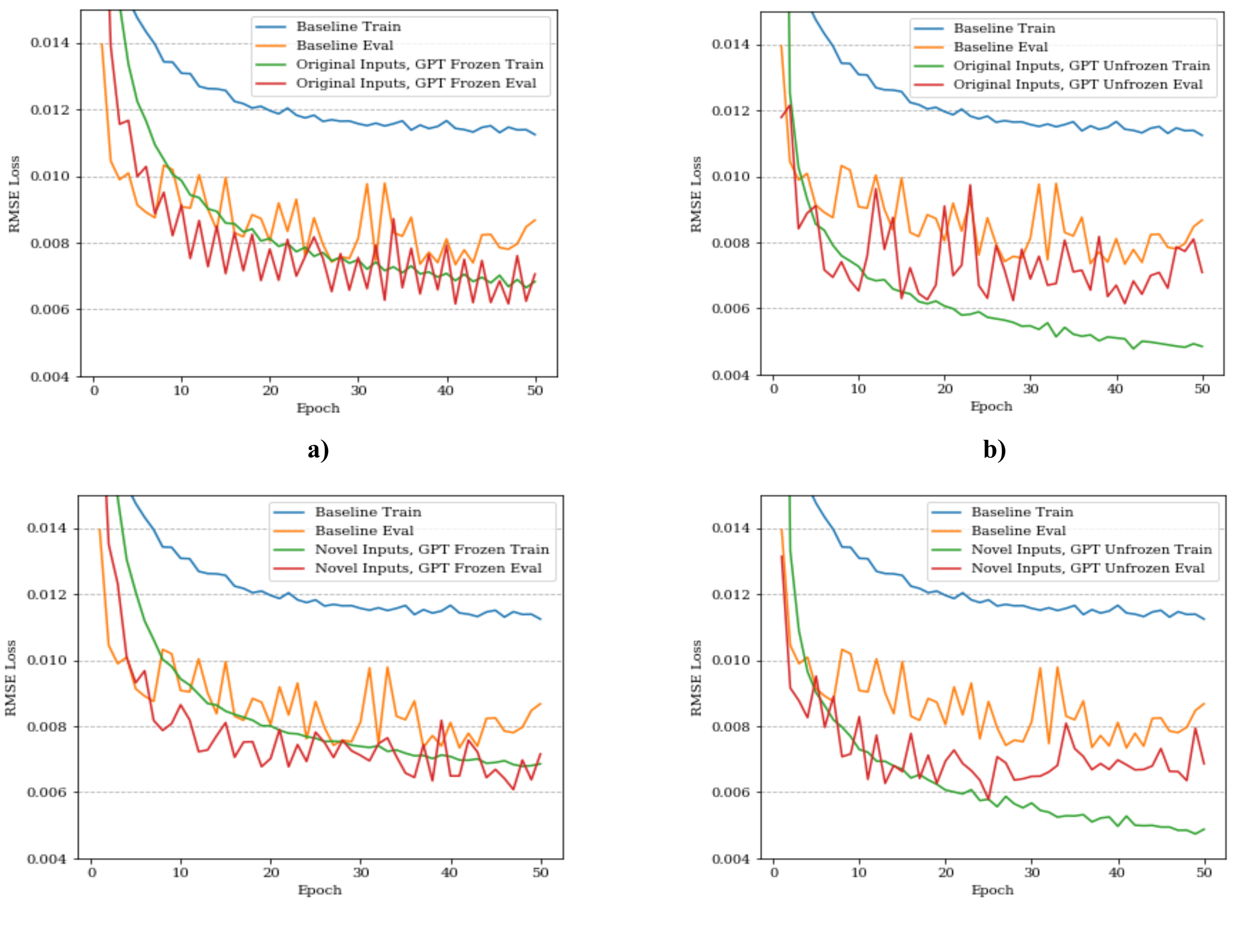

Fig.3: Final Training and Evaluation Results With Tuned Hyperparameters

6.3 Discussion

The tuned hyperparameter values are indicated in section 6.1. For each of these hyperparameters, we made the assumption that if we optimized each of them individually, using the chosen optimized values together would result in the model with near optimal performance. Though this is not entirely true, the assumption allowed us to experiment with more hyperparameter values in parallel.

After tuning the hyperparameters, we performed 4 final experiments to evaluate the performance of our model as shown in Table IV and Fig.4. These experiments can be grouped into 2 categories depending upon whether we used the additional new input features (c, d) or not (a, b). In each of these two categories, we differentiate between the experiments based on whether we use a frozen pretrained GPT-2 model (a, c) or not (b, d). For the experiment a) i.e. with pretrained GPT-2 and without new features, our model performed significantly better than the baseline implementation as shown in Fig. 3a. This validates our choice of hyperparameters, as with these new values we achieve a better training and evaluation loss as compared to baseline model. Also, even though we turned off the dropout layer for the FC layer, this model does not have variance issues as the training and evaluation curves closely follow each other.

To improve the bias of the model, we unfroze all the layers in GPT-2 in experiment b). The results for this experiment were in line with our expectations as we achieved a significantly lower bias - even lower than the LSTM model used in [1]. Since GPT-2 is a very large model, it was better able to learn from our input features. But at the same time we observe in Fig. 3b that it overfit our training dataset leading to high variance starting around epoch 20. Since there was a significant reduction in bias, we believe that training on a bigger dataset might be useful as it will allow us to use the power of transformer models without overfitting the data. Due to paucity of time, we were not able to train on the bigger dataset.

For experiments c) and d), we added our 16 new input features to our dataset. But contrary to our expectations, the results were respectively similar to experiments a) and b). Experiment c) shows a steady decrease in training and evaluation loss, with both decreasing in lockstep with one another. Experiment d) shows GPT-2's great capacity to learn the training dataset with its low training loss, but just like b) there are clear signs of overfitting to the training set starting around epoch 20. We didn't see further improvements in performance neither in training nor in evaluation loss. We believe that this happened because these novel input features were calculated based on the noisy body markers. So they were not noise free and did not add enough novel information to our model to make a performance impact. If we calculate these new features based on actual body measurements, we believe the model can achieve even better accuracy.

7. Conclusion

In this project, we used the GPT-2 transformer architecture on a novel application about body marker augmentation. We further added some new input features based on subject body measurements to improve the RMSE accuracy. Our model achieved the best case RMSE performance of 0.00484653 for training loss and 0.00635123 for evaluation loss, both of which are better performances as compared to the previous work done using an LSTM by Uhlrich et al. (2022) [1]. We believe training on a bigger dataset can help the transformer model learn better and have even better accuracies. We also believe calculating the new input features using actual body measurements should add enough novel information into the model to improve its performance further. Due to limited time, these ideas could not be pursued in this project. However, we hope we can incorporate these modifications into our project later.

8. Contribution

Akshit Goel: Worked on setting up the pretrained GPT 2 model for current application, conducted experiments and documented the results.

Rhythm Patel: Performed various experiments for hyperparameter tuning and unfrozen GPT-2 model, added utility functions to the code, early experiments with transformer architecture by creating model from scratch

Caleb Terrill: Experimented with different transformer models including a Hugging Face Sequence2Sequence model, a Pytorch Transformer, and a Tensorflow Transformer. Wrote the training loop that was eventually used to train the final GPT-2-based model and tune hyperparameters. Generated data graphs.

All the authors worked on designing a detailed framework/flowchart of the project. All authors contributed nearly equally in pre-processing, experiments, tuning of model parameters, and writing of reports.

9. References

[1] Uhlrich, Scott D., et al. "OpenCap: 3D human movement dynamics from smartphone videos." *bioRxiv* (2022).

[2] Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

[3] Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.

[4] Jumper, John, et al. "Highly accurate protein structure prediction with AlphaFold." *Nature* 596.7873 (2021): 583-589.

[5] Carion, Nicolas, et al. "End-to-end object detection with transformers." *European conference on computer vision*. Springer, Cham, 2020.

[6] Petrovich, Mathis, Michael J. Black, and Gül Varol. "Action-conditioned 3d human motion synthesis with transformer vae." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021.

[7] Lu, Kevin, et al. "Pretrained transformers as universal computation engines." *arXiv preprint arXiv:2103.05247* (2021).

[8] Radford, Alec, et al. "Language models are unsupervised multitask learners." *OpenAI blog* 1.8 (2019): 9.

[9] https://huggingface.co/gpt2