Project Report : CS 7643
Motion Prediction

Ibrahim Abdulhafiz Khubaib Ilyas Karla Saillant Matthias Stefan Schnabl
Georgia Institute of Technology
Atlanta, GA
{iabdulhafiz3,kilyas3,ksaillant3,mschnabl3}@gatech.edu

Abstract

Human motion prediction is the task of predicting the next motion of human joints and limbs using the data collected from immediate past motion. This has a wide array of use cases ranging from autonomous vehicles, sports science, surveillance, healthcare and entertainment. Autonomous vehicles use motion prediction to predict pedestrian motion to avoid collision and improve safety. In sport science, it can be used to improve athletic performance and efficiency. This project aims to explore a variety of recent recurrent deep neural network models such as RNN, Seq2Seq, and Transformer which have been proposed. We analyze the performance of existing models and expand on their implementation with the goal of improving their performance and robustness.[1] *Particularly, a custom Transformer Decoder is introduced to mimic the physical constraints of human motion, which show promising results on preliminary trials. As an extension, a split-joint architecture is additionally proposed.*

1. Introduction

1.0.1 Background and Motivation

Accurate motion prediction is crucial for applications in the fields of Robotics and Computer Vision. Enhancing the interactions between humans and computers to improve the accuracy of which human movements are anticipated and responded to, particularly in high safety risk disciplines such as autonomous vehicles, can help lead to safer and more productive environments. Understanding and predicting pedestrian motion is crucial for the safe integration of autonomous vehicles into human-populated environments. Another possible application of human motion prediction can be seen in the entertainment industry. The animation and creation of virtual characters can be used to create real-time realistic life-like characters in video games, movies, and virtual environments. There is a plethora of use cases for motion prediction including but not limited to gesture recognition, sports analysis, and other human-computer interaction applications.

A common challenge in human motion prediction is the synthesis of short-term and long-term motion frames that should follow a given series of frames. This project aims to predict successive frames in a human motion sequence. This is done by observing the previous frames in the motion sequence and applying deep learning methods to predict the next frame. Instead of predicting poses, we tackled this problem by predicting the angle of each joint individually. We suspected that this would help the model learn the relationship between each joints and their preceding motions better. We sought to improve on the SOTA transformer research described in *Spatio-temporal Transformer for 3D Human Motion Prediction* [1]. The Facebook Fairmotion library [4] implemented much of what was discussed in the paper. We used this as our baseline and implemented our own custom transformer-based architecture with the goal of improved accuracy on prediction results.

1.1. Related Work

Extended target tracking has been a subject of interest in tracking and fusion for many years. Several algorithms and filters have been proposed to solve this problem [5]. The most popular approach in recent years has been the use of Random Finite Sets (RFS) and other approaches that stem from RFS [8] [11]. In the recent years, due to advances in deep learning, several artificially intelligent approaches utilizing deep learning have become popular among researchers to tackle motion prediction.

In 2015, specifically for human motion prediction use cases, the use of the Encoder-Recurrent-Decoder (ERD) model was proposed. The ERD model is a recurrent neural network that utilizes nonlinear encoders and decoders before and after its recurrent layers. It is an extension of previous LSTM models. The models showed to have suc-

[1] https://github.com/iabdulhafiz/fairmotion

cessfully handled motion capture (mocap) data across variable subjects and domains avoiding drifts for long periods of time [3]. Since then, other solutions have applied methods for language and vision applications, such as Sequence2Sequence models [10], CNNs [7] and Reinforcement Learning [12].

In 2019, Aksan et al. published their research *Structured Prediction Helps 3D Human Motion Modelling*. In this paper, the authors proposed using Sequence2Sequence, QuarterNet, and RNN models on the H3.6M and AMASS datasets to provide a general solution for the human motion modeling problem. They introduced a structured prediction layer (SPL) consisting of prior knowledge of the human skeletal structure to the neural networks. The SPL breaks up the pose into individual joints which can be used with the architectures. Their results yielded a better performance across the board with the AMASS dataset and an augmented RNN model with a structured prediction layer.

Vanilla RNN-based architectures generally struggle with long term predictions. They “suffer from occasional unrealistic artifacts such as foot sliding” [10] while at the same time “it is often observed that there is a significant discontinuity between the first predicted frame of motion and the last frame of the true observed motion” [7]. Another limiting factor in RNNs is, that they “cannot learn to recover from their own mistakes if they are fed only ground-truth during training.” [10]. Due to the issues regarding long term predictions, Martinez et al. [10] focus their seq2seq approach on short-term prediction. Compared to RNNs, CNNs as “can better avoid the long-term mean pose problem and give more realistic predictions.” [7]. Basing their work off of these findings, Aksan, Kaufmann, Cao, and Hilliges later proposed a new state of the art model using spatio-temporal transformers [1].

1.2. Dataset

For our research, we used the Archive of Motion Capture As Surface Shapes (AMASS) dataset, which is a large database of raw human motion capture datasets. AMASS was created by Perceiving Systems, who specialize in computer vision, computer graphics, and machine learning research with the goal of ”learn[ing] digital humans” [13]. This dataset is larger, richer, and more varied than other human motion collections.

AMASS uses SMPL which is widely used to provide a skeletal representation and a fully rigged surface mesh. It unifies 15 different optical marker based datasets into an easier to use framework. It consists of raw motion capture data converted to mimic realistic 3D human rigged body models including soft-tissue dynamics and hand motion. The dataset consists of over 40 hours of motion data, 300 subjects, and 11,000 motions [9].

The data is developed and owned by the Max Planck Institute of Intelligent System, owner of Perceiving Systems. Max-Planck grants non-exclusive and free of charge license for the use of its data so long as it is for noncommercial purposes. The data was captured by 3D scanning consenting human models. While the dataset identifies motion for different joints of the human body as well as eye motion, it does not identify individuals or population subgroups such as age and gender [6].

The final source data we used was the AMASS-DIP Synthetics 60FPS dataset [6] as it was also used by the Facebook Fairmotion library. This was a fairly large dataset, 18 GB, that strained us in terms of training time. We opted for a larger dataset seeking better model performance. The Fairmotion library referenced this version of the dataset that contains 42 hours of human motion data which is about 5% of the AMASS dataset. We used the Fairmotion data loader for this AMASS DIP distribution as well as their pre-processing script. The pre-processing script loaded the raw data, represented them as tuples of sequence windows and split the data into training, validation, and testing sets. It is important to mention that the preprocessing script utilized Aksan, Kaufmann, and Hiliges original pre-processing steps and data splits from their 2019 paper *Structured Prediction Helps 3D Human Motion Modelling* [2]. The split on the data was 90%, 5%, and 5% for training, validation, and test sets respectively. This dataset preprocessing strategy was also used in their later publication proposing a transformer-based architecture [1].

2. Approach

As mentioned in the introduction section, we used Facebook’s Fairmotion library based off of Aksan et al.’s spatial-temporal transformers research as the benchmark implementation and SOTA models. We utilized the AMASS-DIP Synthetics 60FPS dataset as it was both used by the Fairmotion project and recorded as the most performant and diverse dataset by Aksan and other authors [2]. In addition to this, we implemented the two custom transformers described below. The AMASS dataset was loaded and pre-processed as described in Section 1.2. The same preprocessed data was used for all experiments. We seek to forecast future frames or body poses, given a set of pose sequences. This is a sequence modeling task where the input is 120 poses/frames, the output is 24 poses (400ms), an 2s of motion at 60Hz.[4]. Each architecture was trained for 100 epochs. Another issue we faced was the initialisation of dynamic models for split joint architecture into CUDA. As this was not possible to be done in the fairmotion skeleton code, we had to resort to testing it on CPU.

2.1. Benchmark Architectures

Our benchmark architectures were RNN, Seq2Seq, Transformer, and Transformer Encoder. All architectures

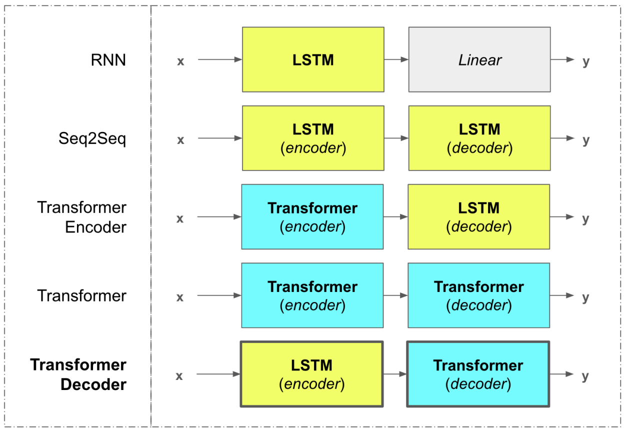

Figure 1. **Deep Learning Architectures** Overview diagram of the high-level architectures used to predict human motion.

require encoding both the spatial and temporal aspects of human motion in order to be able to generate the output sequences. The RNN architecture consisted of an LSTM encoder followed by a linear layer. The Seq2Seq architecture consisted of both an LSTM encoder and decoder. The Transformer architecture consisted of both a transformer encoder and decoder. The Transformer Encoder architecture was comprised of a transformer encoder with an LSTM decoder. See Figure 1 for a visual representation of each architecture.

2.2. Transformer Decoder Architecture

The unique benefits of incorporating sequence to sequence architectures relates to the fact that global context can be acquired before it is decoded sequentially. Following analysis on the aforementioned conventional architectures, we propose the Transformer Decoder architecture which utilizes LSTM as the encoder and a cross-attention transformer as decoder. Inherently, transformers are designed to encapsulate global context throughout the network due to their attention. On the contrary, LSTM networks gradually build the global context of the input space, hence their Long-Short Term Memory.

When analyzing the problem at hand, predicting human pose into the future necessitates attention to all joints. However, when encoding the global context (i.e, the encoder), we propose that the progressive encoding inherent in LSTM would enable the network to learn better. Consequently, the benefits of cross-attention from the decoder would allow for good global correlation during the decoding stage. This inverted approach (opposite to Transformer Encoder) should therefore enable the network to better learn the future prediction of human poses.

2.3. Split Joint Architecture

We implemented a separate RNN module for each body joint. We believed that removing the vectoral dependency between each body joint would help the module to learn the relationship between input motion sequence and the individual body joint better. The architecture is the same as vanilla RNN except that there are n RNN modules - n being the number of body joints we want to track. Each RNN module outputs hidden-dim/n outputs which are then aggregated and fed to the fully connected layer to be then predicted. The architecture is illustrated in Figure 2.

The code for split joint architecture is implemented in the rnn-split-joint-architecture branch of our Fairmotion repository.

2.4. Problems

One of our largest problems was the sheer size of the data and the time it took to train the models. Only one team member had the necessary processing power to run the preprocessing, training, and validation scripts locally, using the Nvidia 3090 GPU with an Intel i9 CPU and 30GB of RAM. Others needed to run the models on Google Colab using an increased System RAM and T4 GPU.

3. Experiments and Results

3.1. Metrics and Loss

We used Mean (Euler) Angle Error, Euler, and Mean Squared Error as our metrics of success. Mean Angle Error and Euler were at 60Hz meaning the poses were at 80, 160, 320 and 400 ms. See Table 1 for the Mean Angle Error results and Table 3 for the Euler results of our benchmark and custom architectures. Figure 3 shows the MSE loss for each architecture at different magnitudes.

As stated by Aksan and other authors, Mean Squared Error can become problematic for auto-regressive models with long time horizons as the ground-truth is often shorter that the predicated horizon [1]. Other complementary metrics, such as Mean Angle Error and Euler, that benchmark in frequencies help us conduct further analysis.

MAE was calculated using the ”Euclidean distance between predicted and reference Euler angles averaged over all joints, summed up to N frames” [4]. We used the Euler angles as a performance metric measured at lower time steps which showed better results. The calculations and conclusions were drawn from Martinez’s work on using RNNs for human motion prediction in 2017 [10]. For this reason, we opted to use the same metrics.

MAE and Euler loss was used to compute both the total training and testing/validation loss and loss per iteration.

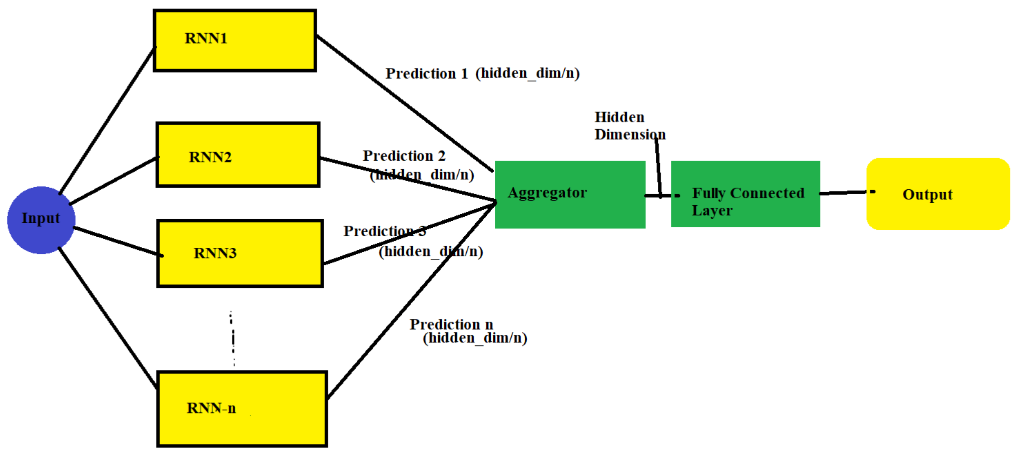
Input
RNN1
RNN2
RNN3
RNN-n
Prediction 1 (hidden_dim/n)
Prediction 2 (hidden_dim/n)
Prediction 3 (hidden_dim/n)
Prediction n (hidden_dim/n)
Aggregator
Hidden Dimension
Fully Connected Layer
Output

Figure 2. Proposed RNN split-joint architecture.

3.2. Benchmark Results

Figure 3 shows the MSE loss results of the four benchmark architectures. Solid lines represent the training loss while dashed lines represent the validation loss. From this graph, we can see that the benchmark Transformer architecture which had a Transformer Encoder and a Transformer decoder had the lowest training loss. In contrast, the RNN architecture containing an LSTM Encoder and a Linear Decoder showed a better validation loss than all others. While the Transformer training loss was better than the validation loss, this does not hold true for the other architectures. Often, having a better training loss than validation loss can indicate that the validation data was not well represented in the training data. Specifically for the Transformer, the model may be underfitting as the validation loss was greater than the training loss. As this phenomena does not occur for the other models, it seems that they are able to process and train on the data well.

The MAE and Euler loss results in Tables 1 and 3 support Aksan and co-author's work. Aksan's papers showed how Transformers and Seq2Seq (LSTM Encoder and LSTM Decoder) outperform vanilla RNN models. In many cases, our Seq2Seq (LSTM based model) slightly outperforms the Transformer models. This detail slightly contrasts Aksan's work where he found that Transformers performed better across the board. This could be due to our hyperparamter choices, especially the epochs and dataset size. The Seq2Seq/LSTM and Transformer models are similar within an accepted deviation.

We opted to run the benchmarks and perform the analysis against the existing models as they are mentioned as the baseline models for the various works. This gave us a comparison for our custom architectures later described in the paper.

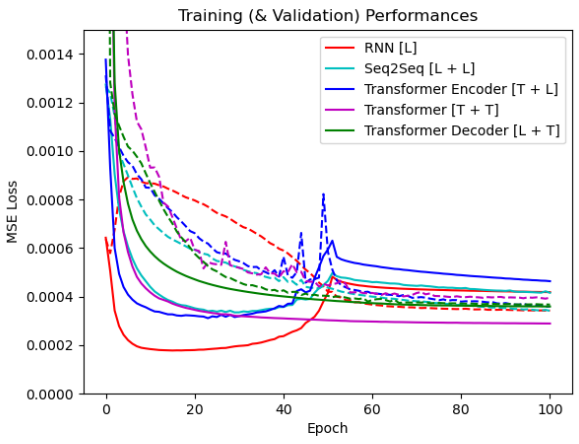

Figure 3. **Training and Validation MSE Loss** The above graph shows how the MSE training (solid-line) and validation (dashed-line) losses compares for all benchmark architectures as well as the custom Transformer Decoder architecture. L refers to LSTM, T represents transformer, and $+$ separates encoder from decoder.

3.3. Transformer Decoder Architecture Results

To begin with, as visible in Fig. 3, all approaches show similar performances to each other and hence it is difficult to claim an absolute winner on the basis of pure metrics. How-

Model	MAE@6	MAE@12	MAE@18	MAE@24
RNN	9.024	8.416	28.255	38.433
Seq2Seq	4.615	10.663	18.078	26.567
Transformer Encoder	5.876	12.428	20.219	28.973
Transformer	4.204	11.827	21.172	31.449

Table 1. **Validation MAE Results** Results for the Mean Angle Error rounded to the third decimal for benchmark architectures.

Model	MAE@6	MAE@12	MAE@18	MAE@24
RNN	8.901	18.428	28.428	38.753
Seq2Seq	4.29	10.111	17.305	25.453
Transformer Encoder	4.181	10.198	17.591	25.902
Transformer	4.615	11.406	19.073	27.360
Transformer Decoder	4.583	11.561	19.532	28.166

Table 2. **Validation MAE Results for adjusted batch and hidden size** Results for the Mean Angle Error rounded to the third decimal for benchmark and custom architectures.

Model	E@6	E@12	E@18	E@24
RNN	1.523	1.596	1.664	1.715
Seq2Seq	0.849	1.108	1.317	1.467
Transformer Encoder	0.989	1.178	1.368	1.515
Transformer	1.030	1.410	1.635	1.754

Table 3. **Validation Euler Results** Results for Euler calculation rounded to the third decimal for benchmark architectures.

Model	E@6	E@12	E@18	E@24
RNN	1.534	1.624	1.691	1.736
Seq2Seq	0.808	1.075	1.278	1.408
Transformer Encoder	0.822	1.114	1.303	1.437
Transformer	1.025	1.199	1.330	1.416
Transformer Decoder	1.042	1.241	1.370	1.477

Table 4. **Validation Euler Results for adjusted batch and hidden size** Results for Euler calculation rounded to the third decimal for benchmark and custom architectures.

ever, when taking into account both training and validation results, the Transformer Decoder appears to achieve the best results. One unique interesting property that is clearly noticeable is the divergence of training and validation loss in the initial phase, then converging in opposite direction, and finally behaving more like a natural loss. This unusual phenomenon in training eventually leads to the validation loss becoming lower than the training loss. The only case this could be possible is if the network has not generalized across the training set and the small validation sample remains an exclusive subset of the training sample. In other words, it proves the network certainly does not represent the entire training sample properly despite being ”superior” on paper based on the validation metrics. For all networks that have an LSTM decoder (RNN, Seq2Seq, and Transformer Encoder), this exact behaviour is visible. Only for the Transformer architecture do the losses behave reasonable. However, despite having the best training loss, the validation loss is the worst.

The distinguishing aspect of our proposed Transformer Decoder is that the loss curve behaves vary reasonable. With the training and validation loss converging to an identical value, this indicates that the network is better capable of generalizing on the entire dataset. Given that the task of future pose estimation is diverse, it is possible that certain architectures may overfit on certain activities and hence show superior performances in some and inferior in others. As for why utilizing a transformer for the decoder, it appears that the global context resulting from the cross-attention enables the architecture to consistently predict future trajectories across diverse activities. This is one of the great achievements of transformers; their ability to generalize very well. However, it is also known the transformers require diverse training samples to train even on small tasks (unlike other forms of neural networks when compared with similar hyper-parameters). The sequentially-temporal structure of LSTM seems to be the best explanation as to why the networks was capable of learning joint features for the encoder phase better than transformers and also why LSTM decoders can overfit on particular tasks way easily without the need for large datasets and training (again, relative to transformers). Hence, the inverted approach regarding selecting the LSTM for the encoder and transformer for the decoder proved to show positive results when analyzing the training in the bigger picture. The benefits of both approaches are combined allowing for better prediction of poses.

3.4. Split Joint Architecture Results

Training the split architecture showed initial promising results. However, due to the extensive training time necessary to get final results, it is not certain how well this approach fairs with the benchmark and our custom Transformer Decoder architecture. It is anticipated that the split architecture paired with the shared loss would force the network to better predict motion semi-independently and hence allow the network to be much more robust to out-of-domain samples (activities not in the training set).

3.5. Hyperparameter Modification

We trained each model twice, once with a batch size of 512 and hidden size of 1024 and another time with batch size and hidden size both being set to 256. Decreasing both parameters lead to significant improvements in both MAE and Euler for all architectures. One reason could be that Smaller batch sizes often lead to better generalization. A smaller batch size introduces more noise into the training process, which can help the model to avoid overfitting to the training data. This can be particularly beneficial when dealing with complex models. The decrease in hidden size might also affect the model's capacity. A smaller hidden size can result in a less complex model, which may generalize better, especially when dealing with limited data. See tables 2 and 4.

4. Conclusion

Human motion prediction entails a high degree-of-freedom and hence remains challenging for neural networks to solve. Indeed, the forecasting nature of motion prediction fundamentally means no direct solution is possible. Due to the temporal nature of this problem, recurrent neural network architectures are best primed to learning accurate predictions of future poses. Beyond the various benchmark approaches, from RNN to Transformers, our custom Transformer Decoder architecture shows promising behaviour when training and validating on the AMASS dataset comprised of various human activities. By taking advantage of the temporally-sequential LSTM encoder and cross-attention of the transformer decoder, a physical explanation exists as to why this structure mimics the interlinked nature of joints (locally; encoded) and the overall action being performed (globally; decoded). Hyperparameter tuning and acquiring alternative datasets would be necessary to fine-tune the architecture and determine any flaws it may have. Beyond the family of sequence to sequence RNN architectures which have been adopted mainly in the language, there certainly exists room for a more complex and domain specific architecture that builds upon the fundamental physics and mechanics of human motion. After all, human motion is strictly governed by the internal and external dynamics of the individual.

5. Work Division

Please see Table 5 for the contributions and delegation of work among team members.

Student Name	Contributed Aspects	Details
Ibrahim Abdulhafiz	Initial Setup, Implementation, and Analysis	Setup Fairmotion project locally. Help guide other team members on how to run benchmark metrics. Created and Analyzed the custom Transformer Decoder architecture.
Khubaib Ilyas	Implementation and Analysis	Setup Fairmotion running in local PC as well as Google Colab. Ran and documented results for motion prediction using RNN. Implemented the split joint architecture in a separate branch for RNN.
Karla Saillant	Initial Google Colab Setup and Analysis	Setup the initial Google Colab environment for the team. Worked on analyzing benchmark results and the effects of using different models. Contributed analysis to various areas in the paper.
Matthias Stefan Schnabl	Implementation and Analysis	Trained various models against different sets of hyperparameters. Analyzed results. Contributed analysis to various areas in the paper.

Table 5. Contributions of team members.

References

[1] Emre Aksan, Manuel Kaufmann, Peng Cao, and Otmar Hilliges. A spatio-temporal transformer for 3d human motion prediction, 2021. 1, 2, 3

[2] Emre Aksan, Manuel Kaufmann, and Otmar Hilliges. Structured prediction helps 3d human motion modelling, 2019. 2

[3] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics, 2015. 2

[4] Deepak Gopinath and Jungdam Won. fairmotion - tools to load, process and visualize motion capture data. Github, 2020. 1, 2, 3

[5] Karl Granstrom, Marcus Baum, and Stephan Reuter. Extended object tracking: Introduction, overview and applications, 2017. 1

[6] Yinghao Huang, Manuel Kaufmann, Emre Aksan, Michael J. Black, Otmar Hilliges, and Gerard Pons-Moll. Deep inertial poser learning to reconstruct human pose from sparseinertial measurements in real time. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 37(6):185:1–185:15, Nov. 2018. 2

[7] Chen Li, Zhen Zhang, Wee Sun Lee, and Gim Hee Lee. Convolutional sequence to sequence model for human dynamics, 2018. 2

[8] Ronald Mahler. Measurement-to-track association and finite-set statistics, 2017. 1

[9] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *International Conference on Computer Vision*, pages 5442–5451, Oct. 2019. 2

[10] Julieta Martinez, Michael J. Black, and Javier Romero. On human motion prediction using recurrent neural networks, 2017. 2, 3

[11] Stephan Reuter and Klaus Dietmayer. Pedestrian tracking using random finite sets. In *14th International Conference on Information Fusion*, pages 1–8, 2011. 1

[12] Borui Wang, Ehsan Adeli, Hsu kuang Chiu, De-An Huang, and Juan Carlos Niebles. Imitation learning for human pose prediction, 2019. 2

[13] Jon Williams. Perceiving systems. 2