Compacting MocapNET-based 3D Human Pose Estimation via Dimensionality Reduction

Ammar Qammaz
ammarkov@ics.forth.gr
FORTH and University of Crete
Heraklion, Crete, Greece

Antonis Argyros
argyros@ics.forth.gr
FORTH and University of Crete
Heraklion, Crete, Greece

ABSTRACT

Abstract. MocapNETs are state of the art Neural Network (NN) ensembles that estimate 3D human pose based on visual input in the form of an RGB image. They do so by deriving a 3D Bio Vision Hierarchy (BVH) skeleton from estimated 2D human body joint projections. BVH output makes MocapNETs directly compatible with a large variety of 3D graphics engines, where virtual avatars can be directly animated from RGB sources and off-the-shelf webcam input. MocapNETs have satisfactory accuracy and state of the art computational performance that, however, prior to this work was not sufficient for their deployment on embedded devices. In this paper we explore dimensionality reduction via the use of Principal Components Analysis (PCA) as a means to optimize their size and make them applicable to mobile and edge devices. PCA allows (a) reduction of input dimensionality, (b) fine-grained control over the variance covered by the maintained dimensions and, (c) drastic reduction of the total number of model/network parameters without compromising regression accuracy. Extensive experiments on the CMU BVH dataset provide insight on the effective receptive fields for densely connected networks. Moreover, PCA-based dimensionality reduction results in a 35% smaller NN compared to the baseline (original NN without any dimension reduction) and derives BVH skeletons without accuracy degradation. As such, the proposed compact NN solution becomes deployable on the Raspberry Pi 4 ARM CPU @ 23Hz.

KEYWORDS

3D Human Pose Estimation, Mobile Devices, VR, Neural Networks, Dimensionality Reduction, MocapNET

ACM Reference Format:
Ammar Qammaz and Antonis Argyros. 2023. Compacting MocapNET-based 3D Human Pose Estimation via Dimensionality Reduction. In *Proceedings of the 16th International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '23), July 05–07, 2023, Corfu, Greece.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3594806.3594841

1 INTRODUCTION

Markerless, vision-based human pose estimation is an important research topic with numerous interesting applications in various fields, such as commerce, automotive, robotics, virtual reality and security and serves as a fundamental building block whenever computer systems need to monitor human subjects. These application domains require accurate 3D human pose estimations which may run effectively and efficiently on mobile first/cloud first computing devices. With the advent of neural networks (NN), the problem of 2D joint estimation from RGB images is now effectively tackled [5, 16]. Gradually, the pose estimation frontier shifted to 3D pose estimation with many methods [36, 37] that continuously pushed the state of the art in terms of accuracy. Subsequently, methods began to also regress 3D mesh data [11, 20] and deal with bodies as well as hands and faces [35] in a holistic way. Most 3D methods focus on accuracy and some of them derive higher-order 3D information [6] like SMPL [22] parameters or BVH [25] skeletons [26, 27].

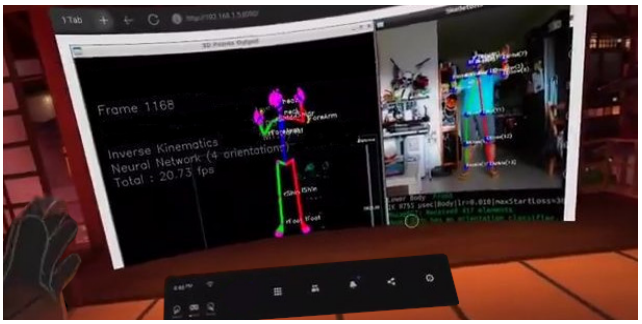

Figure 1: Dimensionality reduction enables 3D body pose estimation suitable for off-the-shelf mobile devices. We experiment using a cheap camera-equipped mobile device, streaming poses through WiFi at interactive framerates to the Meta Quest 2 VR headset. Our work offers a solution for a device that currently lacks this functionality [30].

MocapNET [28] has a number of attractive properties that distinguish it from all other available 3D human pose estimation methods. It can be applied across different types of articulated hierarchies like the upper/lower body [27] or hands [28]. It exhibits theoretical interest due to acting as an inverse NN 3D renderer that derives 3D positions from 2D point clouds. At the same time, it performs Inverse Kinematics (IK) entirely within the neural network in an end-to-end fashion. Furthermore, it achieves state of the art results (see Table 2 of [27]) in terms of computational performance.

Although MocapNETs have numerous advantages, they are currently limited to operating solely on desktop computers. Motivated by the the above stated use-cases and rationale, in this work we

This work is licensed under a Creative Commons Attribution International 4.0 License.

PETRA '23, July 05–07, 2023, Corfu, Greece
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0069-9/23/07.
https://doi.org/10.1145/3594806.3594841

exploit dimensionality reduction to make MocapNETs a viable solution for embedded devices like VR headsets, mobile phones, automotive applications, smart sensors, HCI applications, and a building block for more complex recognition applications.

In spite of the great success of Transformer [33] networks, recent work on Multi-Layer-Perceptron Mixers [31] showcases that overly complicated networks can be counter-productive in-terms of inference speed. This is an additional important motivation for refining MocapNETs that are conceptually similar to MLPMixers. Furthermore, recent jumps in NN parameter counts highlight the importance of NN optimization/pruning as a research topic. For example, even moderate weight pruning yields substantial compute savings in massive networks like GPT-3 [4] that have 1.75×10^{11} parameters due to their immense scale. On the other hand, in smaller networks like MocapNET [28] with 1.3×10^{7} weights, there is simply not the same space left for improvement. Applying a variety of different approaches used in the literature like, model weight-clustering, training-aware quantization and training aware pruning [9], yielded no significant performance improvements in our case. Our observations are similar those reported in works such as [7, 8, 21] and others. Although the exact factors that govern representational capacity and make compression mechanisms effective are not yet completely understood, there is active research [29] to understand, visualize and modify the internal structure of neural networks and recent works prove that "any NN with any activation function can be represented as a decision tree" [2]. These recent findings, prompt us to examine classic ML optimization methods beyond the network itself to increase regression efficiency. We. thus, decide to operate at the input level, exploring dimensionality reduction as a means of understanding and controlling the amount of sample variance contributed by each specific input degree of freedom. Then, we progressively fine tune the number of NN inputs and layer weights so as to control and optimize the network's representational capacity.

A classical tool for dimensionality reduction is Principal Component Analysis (PCA) [1]. We choose to employ it in the context of our work because (a) it preserves the global structure of the data, (b) it does not involve hyper-parameters other than the number of dimensions maintained, and, (c) it is a deterministic algorithm that allows us to decide how much variance of the input data to preserve. Dimensionality reduction can be performed in several other ways. A popular technique is via auto-encoders which are symmetric NNs that feature an information bottleneck which acts as the lower dimensional encoding.

However, such encoders enable no transparency to their internal organization and the identification of the contribution of each of the reduced problem dimensions. What is more, being densely connected, they further degrade parameter count and, thus, performance due to their input size. T-distributed Stochastic Neighbor Embedding (t-SNE) [32] is another option for dimensionality reduction which we do not adopt because of its non-deterministic (randomized) operation, its inability to preserve variance (instead using sample distance) and its maximum of 4 output dimensions. Non Negative Matrix Factorization [18] is not applicable in our case since our input matrices contain negative elements. Similarly, although we attempted to use Dictionary decomposition [23], the size and variance of our data did not allow the algorithm to converge. Finally, Principal Component Analysis offers many variants out of which we selected Full SVD PCA [15] and Randomized PCA [13, 24] omitting other variants like KernelPCA, SparsePCA based on applicability for our data. Finally we also experimented with Factor Analysis [3] and Independent Component Analysis [14] in an attempt to further study how different reduction algorithms impact the solution of our problem.

MocapNETs [26–28] are NN ensembles that deal with the problem of directly regressing a full 3D BVH [25] skeleton from 2D joint estimations detected in an RGB image. The ensembles consist of independent encoders that all share a common input. Each of the encoders regresses a single degree of freedom of the BVH armature. The network training corpus is the CMU BVH Motion Capture dataset [12] which depicts a variety of human actions and motions recorded using a VICON MOCAP system. The problem is very challenging since it involves 2D to 3D regression as well as inverse kinematics, all done in an end-to-end fashion by the neural network. It is also ill-posed since the same 2D joint collection can be explained with more than one 3D skeletons due to ambiguities caused by 3D information loss during image formation.

Direct network training using 2D input is, thus, not feasible. Instead, a variety of 2D descriptors like EDM [19], NSDM [26], NSRM [27] or eNSRM [28] can generate features that networks can learn, in order to successfully tackle the challenging problem of human 3D pose estimation.

2 METHOD

A key design characteristic of any neural network is the number of its network parameters or weights. With a larger number of parameters, a network gains more representational capacity and can accumulate more information during the training session. Higher parameter counts result in larger networks that can tackle more complex tasks. At the same time, these are slower to execute, harder to train and may become prone to over-fitting. In an attempt to retain a small number of weights, convolution operations combined with pooling layers allowed [17] for reducing the aperture of the receptive field of each layer parameter, keeping a smaller number of model parameters that after multiple convolution and pooling layers, gradually learns input global features. However, this technique mostly applies to pictorial input due to the locality of the color/luminance information. There are methods that can regress 3D output from RGB image source. However, MocapNET as well as many other methods use the so-called 2-stage architecture regressing the 3D points from 2D input as a discrete step after extracting the 2D points from an RGB image. The 2-stage architecture enables much richer augmentation during training and compatibility with a wide range of 2D joint estimators. Since the 2-stage problem departs from the original design considerations of convolutional neural networks, it requires a different solution. The design philosophy of MocapNETs is, instead of correlating inputs inside the network, to perform feature correlation directly at the input layer and then just use the effective number of network parameters to regress the output 3D angles. Since all input fields are immediately correlated and there is no spatial locality like RGB input, the network is densely connected (each layer weight is connected to all weights of the subsequent layer). This, however, creates the disadvantage of a

geometric growth of added connections with each additional layer. Combined with the ensemble architecture, this has the potential of creating very large networks. Furthermore, since the operation of the network is opaque, we have no insight on how to trim the input layer (which is the largest) in a meaningful way.

Due to this issue, the original MocapNET formulation proposed the use of a parameter λ (explained in depth in [26]) that acted as a scaling factor on hidden NN layers. This was a coarse solution that did not take into account the input distribution at all. It should be noted that state of the art (in terms of 3D accuracy) methods like Elepose [34] are also attempting similar dimensionality reduction approaches even while following the convolutional paradigm, instead of a densely connected network as the one we use. However, they do not address IK and do not use a 2D joint descriptor but perform PCA on just raw 2D input.

We study the application of dimensionality reduction techniques like Principal Component Analysis (PCA) to the 323 MocapNET input elements of 1.5M available training samples with the goal of reducing and optimizing the parameter count of the network thus making it applicable on mobile devices. Through PCA we gain knowledge of the percent of input variance maintained, despite the lower input parameter count. The observation of the PCA Scree plot (see the two leftmost plots in Figure 2) reveals that just using the first 20 PCA dimensions we can retain 95% of the variance of the network input, 80 PCA dimensions can encode 99.8% of the variation while 210 PCA dimensions encode 99.9999% of that variation. These results show that we can effectively discard a very large amount (more than 35% reduction) of our input since the information it contributes is already present in other inputs. This is a very significant reduction, especially in the context of an expensive densely connected architecture.

However, naïve input reduction has unforeseen consequences, since NNs shrink so much that they no longer maintain the capacity to solve the problem. The baseline method [28] using a $\lambda = 1.0$ [26] created encoders of 152K parameters each, which combined in a full ensemble amount to 5.5M parameters for the upper body and 3.7M for the lower body. Using a PCA basis of 50 dimensions that maintains 98% of input variance, and keeping the network exactly as described in the baseline [28] encoders degenerate to $\sim$ 4.5K parameters, each. For comparison, the first 323 inputs from the first layer to the second feature close to 40K parameters, an order of magnitude more. The concatenation of shrunk encoders creates ensembles of 162K parameters for the upper body and 120K parameters for the lower body. To better understand the extreme degeneration of the network we can think about the following: with naïve input reduction we are essentially trying to tackle the whole 2D to 3D BVH estimation problem using the number of weights previously devoted for a single degree of freedom of the baseline approach. This is clearly not feasible and the NN is unable to correctly respond to input due to its diminished capacity. To remedy NN shrinking and since PCA enables fine tuned control over input and knowledge of its explanatory impact, we also resort to using the λ variable [26]. However, in [26] λ assumed values greater that 1, to shrink the network [26]. In contrast, in this work we set $\lambda < 1.0$ in an effort towards maintaining the NN size, despite the very small input.

3 EXPERIMENTS

We perform dimensionality reduction to input 2D coordinates concatenated by NSRM [28] descriptors. An interesting experiment is to only apply PCA to raw 2D data as done by [34]. Significantly different Scree and clustering plots are observed with NSRMs, as seen in Figure 2, giving rise to a more structured sample order. We show how translation/rotation invariant NSRM descriptors [26, 28] affect samples, highlighting them with color, based on ground-truth d.o.f. and observe that different areas gain spatial coherence compared to raw 2D data. This explains why descriptors simplify the hard task of sample separation making direct NN IK regression feasible.

To provide a broad view of the PCA input compression effect, we vary dimensionality with 210, 180, 150, 120, 90, 50, 30 and 15 PCA dimensions as seen in the first row of Table 1. Baseline method results can be found on columns labeled 323, the non compressed input element number. We observe that 210 PCA dimensions maintain virtually all input variance despite amounting to $\sim$ 35% less input dimensions so we use this as our minimum compression setting. Since networks that have 153K parameters are created by baseline 323 input configurations and they can be reduced by $\approx$ 35% while being certain that the culled inputs are redundant, we are motivated to experiment with networks that have a similar reduction ratio in parameter count (105K parameters). Since naïvely reducing the input leads to aggressive parameter reduction, another experiment consideration is to attempt similarly intense NN compression. Using the λ of the baseline formulation [26] with $\lambda < 1.0$ allows us to maintain a larger number of NN parameters and thus facilitate parametric control for our study. We use 32K parameters which is 80% less than the baseline although still bigger than a naïvely compressed network. We enforce the 32K/105K constant network sizes so that accuracy fluctuations can be clearly attributed to the dimensionality reduction technique and not diminished network capacity. We resort to λ variable [26] scaling to achieve this. For example 15 PCA inputs with $\lambda = 0.048$ create networks with 105K parameters, while the same input and $\lambda = 0.086$, 32K parameters.

3D human pose estimation algorithms are typically benchmarked by calculating mean per joint position error (MPJPE) after Procrustes analysis, which is translation and rotation invariant. Despite regressing higher-order BVH output instead of just 3D points, we perform this experiment since it indicates overall aggregate pose accuracy. We observe that despite the "lossy" compression, the 105K encoder network with 60 PCA input dimensions performs better in 3D Minimum Average Error (M.A.E.) compared to the baseline. This is explained (a) by the fact that the network capacity is directed towards fewer inputs with more explanatory power and (b) because input variance is not spread across more variables. We also see that supplying many redundant dimensions to a relatively small neural network has an adverse impact on learning to derive poses, despite training still having access to the same more useful inputs. An intriguing discovery is that these less significant inputs are not entirely disregarded by back-propagation. Finally, despite managing to recover the 3D pose we observe that the absolute 3D position is not correctly recovered with less than 150 input dimensions (Pos X,Y,Z rows in Table 2). This is an important consideration for applications requiring accurate translation data, since they will need to use a combination of different PCA dimensionalities for different

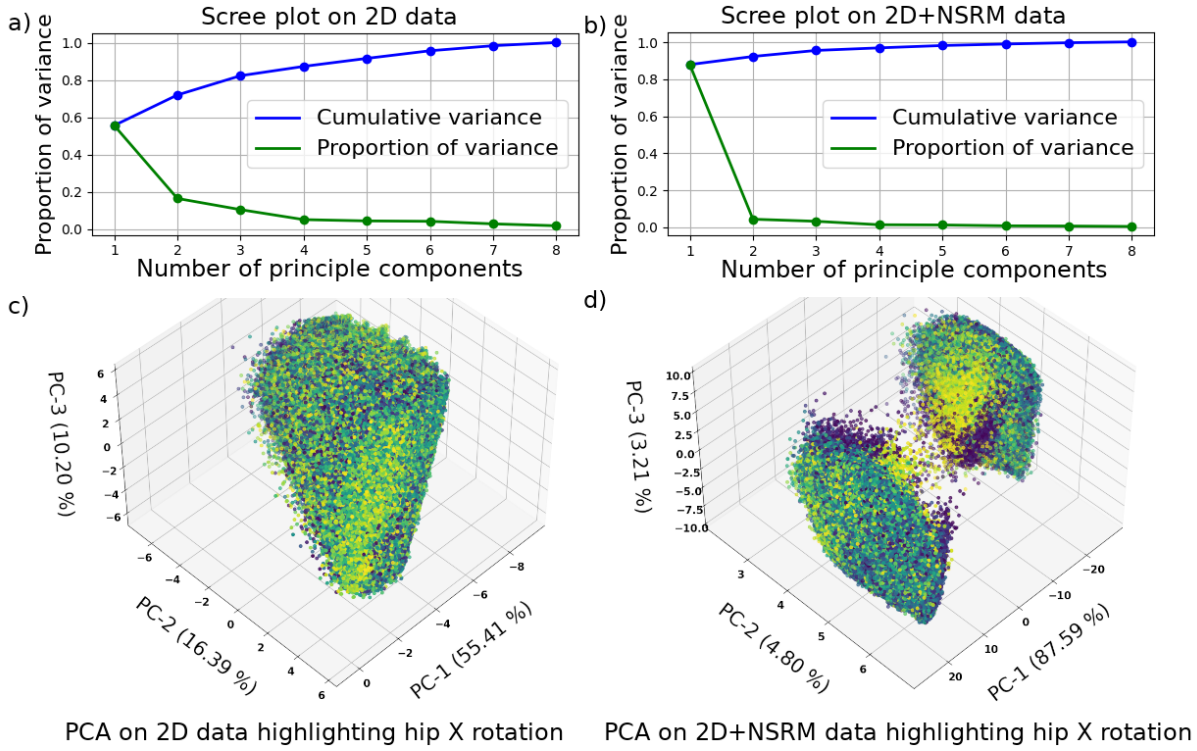

Figure 2: PCA results of 605K CMU BVH [12] pose training samples. From left to right. a) Scree plot using only 2D points as PCA input. b) Scree plot using both 2D points and NSRM matrices. c) 3D clustering of PCA samples using only 2D input with positions corresponding to the 3 most significant PCA dimensions and color highlighting X (pitch) rotation values. d) Same plot using both 2D + NSRM matrices. We observe that the NSRM formulation causes a clearer separation of samples in contrast to raw 2D points.

degrees of freedom of the problem something that is permissible by the extensible ensemble formulation of [28].

Another interesting aspect is training randomization performed in [28]. Both position and orientation are picked from a uniform random distribution during data augmentation. Their entropy is thus much higher than the MOCAP recorded relative joint poses that exhibit regularities due to the constrained range of motion of the human body. We believe that this is the reason why a 35% smaller NN with 60 PCA dimensions surpasses the baseline in M.A.E. since PCA only maintains frequent and important data. On the other hand the high-entropy positional (Pos X,Y,Z) and orientation (Rot Y) d.o.f. need more than 150 PCA dimensions to be effectively regressed as seen in Table 1 and Fig 3.

Since 210 dimensions seem to be enough in terms of representational power and different network sizes (32K/105K) exhibit similar behavior in terms of M.A.E. and St.D., this prompts use of this configuration to benchmark different decomposition methods and study their behaviour at this setting. Table 2 and Figure 3 show that the NN error is adversely affected by other decomposition algorithms compared to SVD/PCA. It should be noted however, that this should not necessarily be interpreted as that the algorithms themselves are not efficient at the dimensionality reduction task, but rather that the selected dimensions are somehow ill-fitted for the problem we examine. Finally, we study dimensionality reduction impact to specific degrees of freedom of the problem since BVH files mainly contain rotational output. Results are presented in Table 1 in rows 6-20. Although maintaining more input variance with bigger NNs predictably performs better, experiments reveal that for applications that do not require a lot of accuracy even aggressively compressed NNs can accommodate the task.

A final finding is that the original λ variable compression proposed for MocapNETs [26] performs worse compared to λ compressed networks with intelligent input dimensionality reduction. For example comparing 323 input parameters that for $\lambda = 1.95/\lambda = 2.7$ create 32K/105K encoders with a PCA basis of 120 input elements we see less error in terms of M.A.E., and can be more accurate in terms of standalone encoder accuracy for 90 dimensions and more.

Utilizing the python TF-Lite backend of Tensorflow, the ensemble with 150 PCA input dimensions achieved an inference rate of 23Hz on a Raspberry Pi 4 (RPi4) device with 2GB of RAM. Given that flagship smartphones commonly feature double the number of cores clocked at higher frequencies compared to the relatively low-power RPi4 Broadcom BCM2711 Quad core Cortex-A72 SoC, the work we presented now makes MocapNET a viable 3D Pose estimator for mobile devices.

PCA	15		30		60		90		120		150		180		210		323		
#Par.	32	105	32	105	32	105	32	105	32	105	32	105	32	105	32	105	32	105	153
M.A.E.	86.3	85.8	83.9	81.8	82.9	**80.1**	86.2	82.6	88.4	84.7	91.3	83.6	103.2	100.8	100.8	100.9	89.0	91.9	89.0
St.D	35.3	35.7	35.2	**34.9**	37.1	37.0	39.1	38.5	40.4	43.8	47.5	43.7	57.3	58.2	52.3	52.1	46.7	46.7	46.7
PosX	158.2	154.9	152.9	147.0	139.3	132.2	130.0	119.7	37.6	31.1	19.3	15.8	18.7	14.6	16.8	**13.1**	20.3	15.9	15.0
PosY	59.6	58.9	58.5	57.0	55.2	53.2	53.7	51.1	48.2	44.9	7.7	6.1	7.2	5.5	6.7	**5.2**	7.6	6.2	5.9
PosZ	48.7	45.4	47.4	43.9	27.3	25.1	26.5	23.9	24.1	20.8	19.2	16.3	17.3	14.6	15.7	**12.4**	18.4	15.3	14.7
RotZ	12.9	12.7	12.6	12.3	12.1	12.0	12.0	11.8	11.1	10.6	10.3	9.4	10.3	9.4	10.2	**9.2**	10.7	9.7	9.5
RotY	54.6	53.3	51.7	49.8	46.0	43.9	43.1	41.0	26.4	24.5	23.5	21.5	23.1	21.4	22.0	**20.4**	25.8	23.6	23.4
RotX	15.5	15.1	14.9	14.6	14.4	14.1	14.2	13.8	13.4	12.8	12.1	11.3	12.0	11.1	11.5	**10.6**	12.9	11.7	11.4
Z/RS	16.6	15.9	15.4	14.6	14.7	13.8	14.5	13.6	14.2	13.1	14.0	12.8	13.7	12.5	13.4	**12.4**	14.4	13.4	13.1
X/RS	16.9	16.2	15.8	14.9	15.2	13.9	14.8	13.7	14.2	13.1	13.8	12.6	13.6	12.4	13.5	**12.3**	14.6	13.4	13.0
Y/RS	17.1	16.1	15.9	14.7	15.0	13.5	14.3	12.9	13.5	12.1	13.2	11.7	13.0	11.2	12.7	**11.1**	14.2	12.5	12.0
Z/RE	5.1	5.0	4.7	4.5	4.5	4.3	4.4	4.2	4.3	4.1	4.3	4.0	4.2	4.0	4.2	**4.0**	4.4	4.2	4.1
X/RE	7.9	7.6	6.9	6.6	6.6	6.3	6.5	6.2	6.3	6.0	6.2	5.9	6.2	5.8	6.2	**5.8**	6.4	6.1	6.0
Y/RE	13.4	12.8	11.2	10.6	10.6	9.9	10.3	9.6	9.9	9.2	9.7	8.9	9.5	8.7	9.6	**8.7**	10.2	9.1	9.2
Z/LT	12.9	12.6	12.5	12.1	12.1	11.7	11.7	11.3	12.1	10.9	11.1	10.7	10.9	10.5	10.7	**10.2**	11.6	11.1	11.1
X/LT	13.3	12.9	12.5	12.0	11.8	11.4	10.9	10.1	12.0	7.1	7.5	6.6	7.3	6.4	7.0	**6.0**	9.7	8.4	8.2
Y/LT	15.6	15.3	15.2	14.9	14.8	14.3	14.1	13.7	14.9	13.1	13.3	12.9	13.2	12.8	13.1	**12.6**	14.0	13.4	13.4
Z/LK	8.3	7.9	7.2	6.9	6.8	6.1	6.6	6.1	6.6	5.6	5.6	5.4	5.6	5.3	5.4	**5.3**	6.2	6.1	5.7
X/LK	11.1	10.5	9.3	8.7	8.5	7.8	8.0	7.6	8.5	6.4	6.7	6.1	6.9	6.0	6.4	**5.9**	7.3	6.8	6.7
Y/LK	8.0	7.9	7.3	7.2	7.0	6.6	6.6	6.6	6.7	6.2	6.1	6.1	5.9	5.9	5.9	**5.8**	6.5	6.5	6.3

Table 1: Ensemble/encoder accuracy analysis, when comparing regression results against ground truth. Column pairs correspond to a fixed number of PCA dimensions. We maintain a specific network parameter size (#Par., in thousands of parameters) with each column depicting a specific parameter count. For each PCA dimension/parameter count combination, error values presented are in millimeters for rows M.A.E. and St.D. (3D mean average error and standard deviation for 3D joint locations of neck, shoulders, elbows, wrists, hips, knees, ankles) after Procrustes analysis [10] for validation sets 01-10 CMU BVH [12]. The rest of the values showcase training errors and different observations we make. Position X, Y, Z errors show the raw encoder translation output error in centimeters. We observe that maintaining less than 150 PCA dimensions causes networks to no longer accurately regress skeleton position, despite an overall better relative pose regression than the baseline method. The rest of the rows depict encoder rotation errors for each joint degree of freedom in Euler degrees. Rot X, Y, Z is the absolute skeleton rotation. The rest of the abbreviations are : RS: Right Shoulder, RE: Right Elbow, LT: left thigh, LKn: Left Knee. Columns 323 use the full input without PCA in an attempt to compress using MocapNET λ values [26] to the same size as PCA experiments (32K/105K). The baseline (153K/$\lambda = 1$) is also included for a complete comparison. Despite the configuration of 210 PCA dimensions / 105K parameters achieving better training errors in all degrees of freedom its validation procrustes M.A.E. is higher than the best configuration. Aggressive 60 PCA dim/105K param compression achieves better M.A.E. if an application is not concerned about positional components of the recovered pose, this is the case with VR applications.

4 CONCLUSIONS

We successfully reduce the size of the MocapNET NN [28] while marginally improving its accuracy in terms of Procrustes M.A.E [10] by applying dimensionality reduction on its input. We achieve NN weight savings from 25% up to 80% depending on size/accuracy trade-offs. Extensive experiments summarized in Table 1 reveal sweet-spots on configurations that use 105K params and 60/150 PCA dims. We also open the way to 32K encoder networks that can effectively tackle aspects of the problem while being much easier to compute. Despite high-dimensional pose space and densely connected neural networks being opaque, we gain insight on the impact of various input and network sizes. PCA allows us to visualize the high-dimensional input and observe the improved sample coherence when utilizing NSRM descriptors (Figure 2). This newfound knowledge enables creation of compressed MocapNETs deployable on embedded devices. This in turn, enables user perception on mobile devices for 3D avatars in VR, automotive applications, robotics, human computer interaction and more.

ACKNOWLEDGMENTS

This research work was partially supported by BonsApps (EU H2020 Grant no.101015848) AI Talent grant (Winner No. Bons_1OC_20). It was also partially supported by the Hellenic Foundation for Research and Innovation (HFRI) under the HFRI PhD Fellowship grant (Fellowship Number: 1592) and by HFRI under the "1st Call for HFRI Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment", project I.C.Humans, number 91.

REFERENCES

[1] Hervé et al. Abdi. 2010. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics* 2, 4 (2010), 433–459.

Dimensionality Reduction Method	Dims	Min	Max	Mean	Median	Variance	Std. Dev.
PCA with full SVD [15]	150	14.0	300.4	83.6	76.0	1906	47.5
PCA with full SVD [15]	210	11.2	361.8	100.9	93.0	2717	52.1
PCA randomized [13, 24]	150	**9.7**	345.9	112.4	107.3	2249	47.4
PCA randomized [13, 24]	210	10.9	390.3	126.3	124.8	4561	67.5
Fast ICA [14]	150	9.8	364.4	137.7	138.6	2988	54.7
Fast ICA [14]	210	14.1	438.4	141.9	131.3	4854	81.0
Factor Analysis [3]	150	10.7	414.5	155.6	158.1	4742	68.9
Factor Analysis [3]	210	10.8	405.4	153.4	161.9	5323	73.0
PCA with full SVD [15]	60	16.2	314.9	**80.1**	**72.3**	**1370**	**37.0**
Baseline/No Dim. Reduction	323	14.8	**298.7**	91.9	80.7	2185	46.7

Table 2: Analysis of the effect of different dimensionality reduction techniques in terms to rotation/translation invariant procrustes M.A.E. on the problem when fixing the number of kept input dimensions to 150 or 210 (from the original 323), the number of network parameters to 105K and training until early stopping is activated to ensure a maximum optimisation budget. Values represent M.A.E. in millimeters after 3D Procrustes Analysis [10] on the test set. We observe that SVD/PCA behaves best compared to other methods and that the same method with 60 input dimensions surpasses the baseline that however is still better than all 210 dim alternatives.

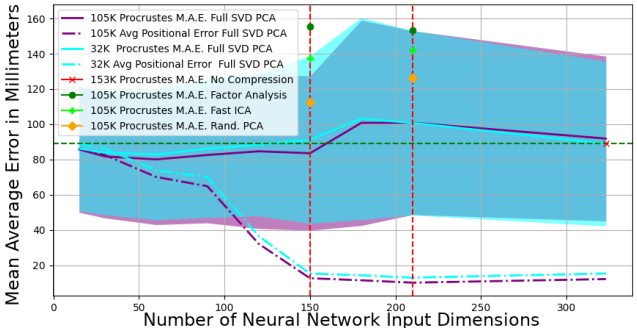

Figure 3: M.A.E. plot for experiments summarized in Tables 1 and 2. Colored area represents standard deviation w.r.t mean error for 105K and 32K SVD/PCA networks of various input dimensionalities. The 210 dim. (vertical red line) configuration is used as the non SVD/PCA experiment basis in Table 2. We observe a complex landscape where although smaller (32K) networks consistently perform worse than larger (105K) ones for < 150 PCA dimensions, as less essential PCA dimensions are added they increase the difficulty of the regression task. We manage to perform better than the baseline for configurations under the green horizontal line despite the reduced parameter counts.

[2] Caglar Aytekin. 2022. Neural Networks are Decision Trees. https://doi.org/10.48550/ARXIV.2210.05189

[3] David Barber. 2012. *Bayesian reasoning and machine learning. Algorithm 21.1.* Cambridge University Press.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. https://doi.org/10.48550/ARXIV.2005.14165

[5] Zhe et al. Cao. 2017. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In *CVPR*.

[6] Sai Kumar et al. Dwivedi. 2021. Learning To Regress Bodies From Images Using Differentiable Semantic Rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 11250–11259.

[7] Jonathan Frankle, David J Schwab, and Ari S Morcos. 2020. The early phase of neural network training. *arXiv preprint arXiv:2002.10365* (2020).

[8] Jonathan et al. Frankle. 2020. Pruning neural networks at initialization: Why are we missing the mark? *arXiv preprint arXiv:2009.08576* (2020).

[9] Google. 2022. *Tensorflow Model Pruning comprehensive guide.* https://www.tensorflow.org/model_optimization/guide/pruning/comprehensive_guide.

[10] John C Gower. 1975. Generalized procrustes analysis. *Psychometrika* 40, 1 (1975), 33–51.

[11] Rıza Alp et al. Güler. 2018. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7297–7306.

[12] B. Hahne. 2010. *The Daz-friendly BVH release of CMU motion capture database.* https://sites.google.com/a/cgspeed.com/cgspeed/motion-capture/daz-friendly-release.

[13] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review* 53, 2 (2011), 217–288.

[14] Aapo Hyvarinen. 1999. Fast and robust fixed-point algorithms for independent component analysis. *IEEE transactions on Neural Networks* 10, 3 (1999), 626–634.

[15] Ian T Jolliffe. 2002. *Principal component analysis for special types of data.* Springer.

[16] Sven et al. Kreiss. 2019. PifPaf: Composite Fields for Human Pose Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[17] Yann et al. LeCun. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[18] Daniel D Lee and H Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401, 6755 (1999), 788–791.

[19] Subhash Lele and Joan T Richtsmeier. 1991. Euclidean distance matrix analysis: A coordinate-free approach for comparing biological shapes using landmark data. *American journal of physical anthropology* 86, 3 (1991), 415–427.

[20] Kevin et al. Lin. 2021. End-to-end human pose and mesh reconstruction with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1954–1963.

[21] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2018. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270* (2018).

[22] Matthew et al. Loper. 2015. SMPL: A skinned multi-person linear model. *ACM transactions on graphics (TOG)* 34, 6 (2015), 1–16.

[23] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. 2009. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*. 689–696.

[24] Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. 2011. A randomized algorithm for the decomposition of matrices. *Applied and Computational Harmonic Analysis* 30, 1 (2011), 47–68.

[25] Maddock Meredith, Steve Maddock, et al. 2001. Motion capture file formats explained. *Department of Computer Science, University of Sheffield* 211 (2001), 241–244.
[26] Ammar et al. Qammaz. 2019. MocapNET: Ensemble of SNN Encoders for 3D Human Pose Estimation in RGB Images. In *British Machine Vision Conference (BMVC 2019)*. BMVA, Cardiff, UK.
[27] Ammar et al. Qammaz. 2021. Occlusion-tolerant and personalized 3D human pose estimation in RGB images. In *IEEE International Conference on Pattern Recognition (ICPR 2020), (to appear)*.
[28] Ammar et al. Qammaz. 2021. Towards Holistic Real-time Human 3D Pose Estimation using MocapNETs. In *BMVC 2021*. BMVA.
[29] Atefeh Shahroudnejad. 2021. A survey on understanding, visualizations, and explanation of deep neural networks. *arXiv preprint arXiv:2102.01792* (2021).
[30] Paul Tassi. 2022. *Mark Zuckerbergs metaverse legs demo was staged with motion capture*. Forbes. https://www.forbes.com/sites/paultassi/2022/10/14/mark-zuckerbergs-metaverse-legs-demo-was-staged-with-motion-capture/.
[31] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. 2021. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems* 34 (2021), 24261–24272.
[32] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
[33] Ashish et al. Vaswani. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
[34] Bastian et al. Wandt. 2021. ElePose: Unsupervised 3D Human Pose Estimation by Predicting Camera Elevation and Learning Normalizing Flows on 2D Poses. https://doi.org/10.48550/ARXIV.2112.07088
[35] Donglai et al. Xiang. 2019. Monocular total capture: Posing face, body, and hands in the wild. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10965–10974.
[36] Ailing et al. Zeng. 2020. Srnet: Improving generalization in 3d human pose estimation with a split-and-recombine approach. In *ECCV*. Springer, 507–523.
[37] Ce et al. Zheng. 2021. 3d human pose estimation with spatial and temporal transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 11656–11665.