MASTER THESIS

Human Pose Estimation using Deep Generative Models

Author:
Lucas MOUROT

Supervisors:
Babak FALSAFI[1]
Pierre HELLIER[2]
François LE CLERC[2]
Cédric THÉBAULT[2]

[1] ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

[2] INTERDIGITAL, INC

August 14, 2019

Contents

Abstract

In this thesis we investigated a variation of deep generative models as a solution to Human Pose Estimation (HPE) from monocular image sequences. Instead of learning the direct mapping from image sequences to human poses, a generative model allows to explicitly learn a latent representation of the human motion distribution in an unsupervised manner. Our model makes it possible to apply the same generic approach to different human skeleton models that traditionally would require very different network architectures. Furthermore, the latent representation allows to refine and complete the human pose, as well as upsample the joints.

1 Introduction

The problem of human pose estimation generally consists of estimating 3D joints position from monocular 2D image(s). It is an ill-posed problem mainly due to the estimation of 3D pose from 2D information which is underconstrained. Further issues include incompleteness of information such that out of field limb(s), occlusions and self-occlusions, limb foreshortening due to the viewpoint or ambiguities when limbs are interlaced. The human pose is usually represented by a joint skeleton whose topology is fixed, typically 12 to 16 joints. Other variants such as multi-person pose estimation or body shape estimation are not considered here.

To our knowledge, most recent approaches to 3D human pose estimation break down the problem in some way or another into 2D pose estimation and then depth regression, since directly estimating the 3D pose is considered too hard. Providing a complete and detailed high quality 2D pose is then crucial to correctly address the problem of 3D human pose estimation. In this work we leverage an existing state-of-the-art solution to 2D human pose estimation from monocular image (temporal) sequences to 2D pose (temporal) sequences by intermediately refining and completing the pose sequence and upsampling the skeleton topology. This allows for the subject that is captured to be partially occluded or partially out of field while still having a human pose estimated spatially and temporally coherent as a whole.

Neural networks having impressive performance in image processing, and in particular deep generative models being highly competitive on image inpainting problems in recent years, we formulated our intermediate process of 2D human pose completion and upsampling as a problem of pose sequences inpainting. We designed a generative model that learns the human motion distribution and derived a method to inpaint pose sequences using our generative model.

Currently the most prominent approaches to produce new samples from high dimensional distributions are variational autoencoders (VAEs) [21] and generative adversarial networks (GANs) [12]. Both have strengths and weaknesses. VAEs are known to be quite simple to train but also to give blurry generated outputs due to oversimplified posterior distribution (Not being able to parametrize a complex distribution, it is often a unimodal distribution like an isotropic gaussian). On the opposite, GANs produce sharper outputs than VAEs but are far harder to stabilize during training probably due to their adversarial nature. Moreover VAEs (and more generally autoencoders) provide a mapping from sample space to the latent representation while GANs provide an estimate of the intrinsic quality of a sample (i.e. the discriminator).
Beyond AEs and GANs, researchers ([22], [23], [24], [25]) have shown that a combination of AE(s) and GAN(s) can yield an efficient generative model, keeping some

advantages from both. Inspired from these rich deep generative networks, we designed a generative model that fuses an autoencoder with a GAN where the autoencoder's decoder is also the GAN's generator. Our model succeeds in keeping the stability of an autoencoder at training time while producing sharp pose sequences and has both an encoder and a discriminator that are crucial for our task, namely inpainting of pose sequences, as explained in detail in section 5.

2 Related Work

2.1 2D Human Pose Estimation from Images

With the introduction of *DeepPose* in [2], research on human pose estimation shifted from classic approaches to deep networks.
Introduced in [3], *Stacked Hourglass Networks* is a pure deep learning based multi-stage approach that capture information at all scales using a repeated bottom-up, top-down processing, by going from full resolution to low resolution and back to full resolution through convolutional layers. The network takes as input 2D images while producing a set of probabilistic 2D heatmaps for each joint. In order to avoid vanishing gradients through the deepness of the stacked hourglasses, a loss term for each hourglass prediction is included in the overall loss. State-of-the-art approaches to 3D human pose estimation often reuse stacked hourglass networks as 2D pose estimation module.

2.2 3D Human Pose Estimation from Images

Beyond general issues in human pose estimation, the 3D version of the problem is in practice even more challenging, due to the lack of training data since existing datasets are either in the wild images with 2D annotations or images produced in laboratory with 3D annotations.
The approach described in [7] gives a way to solve this problem in addition to achieve competitive results on both 2D and 3D benchmarks. They proposed a weakly-supervised transfer learning method that mixes 2D and 3D annotations in a unified network which augments a state-of-the-art 2D pose estimation sub-network (*Stacked Hourglass Networks* [3]) with a 3D depth regression sub-network. They trained both sub-networks end-to-end and exploits the correlation between the 2D pose and depth estimation sub-tasks. In this way, the 3D pose annotations in controlled lab environments are transferred to in the wild images. Finally, they regularize the 3D pose prediction using a 3D geometric constraint making the 3D pose prediction from in the wild images effective in the absence of ground truth depth annotations. This geometric constraint penalizes the pose configurations that violate the consistency of bone-length ratio priors. This work is surely one of the best approaches to 3D human pose estimation in the wild from a single image.
Among other methods that do not split the problem into 2D pose estimation and depth regression, interesting and competitive ones exist and undoubtedly include the structured prediction approach defined in [6]. It consists of a deep learning architecture that relies on an autoencoder (AE) to learn a high-dimensional latent 3D pose representation and accounts for joint dependencies. The autoencoder is pretrained. Then, a Convolutional Neural Network (CNN) is trained to map 2D images into the latent representation learned by the autoencoder while the pretrained decoder map the latent representation back to the 3D pose.

2.3 3D Human Pose Estimation From Image Sequences

The major drawback at estimating the pose from a single image is the resulting temporal inconsistency when applied to a sequence of images. However, while temporal consistency is theoretically easier to reach with pose sequences estimation from image sequences, video chunk inputs are too large to be handled by deep neural networks. Different approaches exist to address this issue and often imply to first estimate the pose for each image of a sequence independently and then processing pose sequences that have much smaller dimensionality than image sequences.
As the logical continuation of the weakly-supervised approach to 3D human pose estimation in the wild [7], Dabral et al. [8] estimate pose sequences from image sequences (each image independently using [7]) and then temporally harmonize the pose estimations using a simple fully-connected temporal sub-network that exploits temporal and structural cues present in predicted pose sequences. They also proposed two additional geometric losses penalizing respectively illegal bone angles and bone length symmetry (left/right) violations.
In contrast with [8], Lin et al. [9] capitalized on Recurrent Neural Networks (RNNs) to exploit sequence-dependent temporal context and on multi-stage refinement, such as in [3]. They proposed a Recurrent 3D Pose Sequence Machine (RPSM) that is used in a multi-stage manner. The RPSM is composed of a 2D pose estimation module, a recurrent 3D pose estimation (LSTM-based) module regressing 3D poses and a features adaptation module inbetween. A sequence of images is thus passed image by image through multiple RPSMs, whose hidden states of LSTM units capture the temporal contextual dependency, in order to refine the previous predicted pose at each stage.

2.4 Adversarial Approaches to Human Pose Estimation

Significant improvements have achieved in human pose estimation with the strategy of regressing heatmaps of each body part using CNN. However, it is difficult to include structural priors of the human body into these networks. Consequently, in some situations, for instance in case of heavy occlusions, standard CNNs-based methods tend to perform poorly. Learning the real human body joints distribution is then crucial, however, explicit learning of such a distribution can be very difficult. Fortunately, deep adversarial approaches, and in particular GANs, are quite efficient to implicitly learn such distributions.
To our best knowledge, Chen et al. [10] are the first to use Generative Adversarial Networks to exploit the constrained human-pose distribution for improving human pose estimation. Indeed, they designed a multi-task generator to predict 2D pose heatmaps as well as 2D occlusion heatmaps from images, and two discriminators, one telling whether the predicted pose is geometrically reasonable while the second discriminates the high-confidence predictions from the low-confidence predictions. Training of their network follows the strategy of conditional GAN (CGANs).

More recently, Yang et al. [11] also proposed an adversarial approach to human pose estimation where the generator is the pose regressor but they reused the work of Zhou et al. [7] for the design of the generator and thus they proposed an adversarial solution to 3D human pose estimation. In addition, instead of having two different discriminator, they designed a multi-source discriminator whose inputs are triplets made of a 2D image and corresponding 2D pose heatmaps and 3D pose, which helps to enforce the pose estimator to generate anthropometrically valid poses even with images in the wild.

2.5 Inpainting Using Generative Models

To our best knowledge, we are the first to address the task of 2D human pose inpainting using deep generative networks. our approach is related to the existing semantic image inpainting framework designed in [26]. Indeed Yeh et al. proposed to recover the missing portion of an image using a trained GAN by searching for the closest encoding of the corrupted image in the latent image manifold using a contextual loss and a prior loss. The context loss minimized differences between input and generated images on the available input pixels while the prior loss penalizes unrealistic images by maximizing the discriminator score on the generated image.

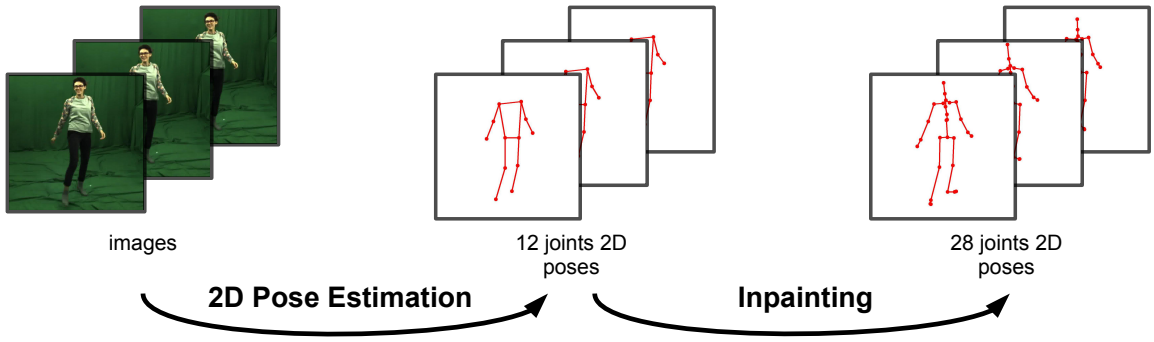


Figure 1: Illustration of the proposed additional stage to 2D human pose estimation, i.e. inpainting.

3 Contribution to Human Pose Estimation Pipeline

In this work we propose to add a step of inpainting to the traditional human pose estimation pipeline as depicted in figure 1. It takes place after a first 2D pose estimation and is intended to correct and increase the details of the pose estimated only using human motion priors.

3.1 2D Joints Position Estimation: *Alpha Pose*

We chose to rely on *Alpha Pose* [1] for the first step in our workflow, that is the estimation of 2D joints position sequences from monocular image sequences. *Alpha Pose* is an accurate pose estimator which is the first real-time open-source system that achieves 70+ mAP (72.3 mAP) on COCO dataset [28] and 80+ mAP (82.1 mAP) on MPII dataset [27], where mAP stands for Mean Average Precision which is the common measure in multi-person pose estimation since it penalizes for false positives that are not a part of the ground truth. They also provide an efficient online pose tracker called *Pose Flow* based on [5] that is the first open-source online pose tracker that achieves both 60+ mAP (66.5 mAP) and 50+ MOTA (58.3 MOTA) on PoseTrack Challenge dataset [30].

3.2 2D Pose Sequences Inpainting

Since *Alpha Pose* provides a 2D pose made of 12 joints (see topology in figure 6a) whose potentially some joints are missing, here we aim at completing, upsampling and adjusting the pose as a whole. The basic idea is to find the latent code that best reconstructs the input when passed through the generator. The inpainted pose is

[1] Implementation based on [4] and [5], and available at `https://github.com/MVIG-SJTU/AlphaPose`

then directly computed from the latent code found, using the generator.
More formally, let $Z \subset \mathbb{R}^L$ be the latent space of dimensionality L and $X \subset \mathbb{R}^{F \times J \times 2}$ the space of 2D pose sequences, where F is the sequences length and J is the number of joints per skeleton. Then the generator can be abstracted as the mapping $G : Z \to X$ from the latent space to the space of 2D pose sequences. Given an input pose sequence $x \in X$ to be inpainted, we can formulate the inpainting of x as the task of finding the latent code z^* such that

$$z^* = \arg\min_{z \in Z} \mathcal{L}(x, G(z))$$

where $\mathcal{L} : X^2 \to \mathbb{R}$ is a metric to be defined. From this optimal latent code z^* we can generate the 2D pose sequence $x^* = G(z^*)$ that optimally reconstructs the input x. Considering $\mathcal{L}$ as a loss function, this minimization problem would be solved by a gradient descent in the latent space.
Technical details such as the definition of the loss or the latent code initialization are highly dependent of the generative model and are discussed in section 5.

3.3 3D Pose Regression from 2D Pose

Since most state-of-the-art solutions to 3D human pose estimation break down the problem into 2D human pose estimation and then depth regression, the additional inpainting step we propose could be used for 3D human pose estimation as well. In our case we focused on 2D poses and left the 3D case for future work.

4 Deep Network Architecture

4.1 Deep Generative Networks Building Blocks

Here we first introduce the basics of deep generative networks.

Autoencoders (AEs) An autoencoder is a type of neural network used to efficiently learn a latent representation of a set of data in an unsupervised manner. It is composed of two sub-networks, an encoder and a decoder, with the latent code in between. Generally, the latent space has a lower dimensionality than the sample space which also enforces the network to learn an efficient dimensionality reduction through the encoding. An autoencoder is trained by minimizing the difference between its input and its output.
Several variants exist, with the aim of enforcing some useful properties in the latent representations learned, including Denoising Autoencoders [20] where the encoder is fed with corrupted (noisy) input while trying to reconstruct the clean input, which makes the model robustly extracting features from inputs, and providing denoised outputs.

Variational Autoencoders (VAEs) Unlike others autoencoders, Variational Autoencoders are generative models. Despite the fact they are classified among autoencoders, their mathematical formulation differs significantly. VAEs make strong assumptions about the distribution of the latent representation, i.e. that the data is generated by a directed graphical model $p_\Theta(x|z)$ and that the encoder is learning an approximation to the posterior distribution $q_\Phi(z|x)$, where x is the data sample, z its corresponding latent representation and Θ and Φ are respectively the parameters of the encoder and decoder. VAEs are known to be quite simple to train but have generally the disadvantage to generate blurry samples due to the strong (often Gaussian) assumptions on the posterior distribution.

Generative Adversarial Networks (GANs) Typically, a GAN is composed of two sub-networks, a generator and a discriminator. The generator is trained to produce realistic samples from the latent distribution while the discriminator aims at distinguishing real samples from samples generated by giving an estimate of the probability that an input sample is real. Both are trained adversarially, competing against each other. GANs are often viewed from a game theory point of view: the discriminator and the generator are trained simultaneously to find a Nash equilibrium to a two-player non-cooperative game. Although theoretically elegant, the convergence of GANs is in practice difficult to obtain and very sensitive to hyperparameters, at least partially due to the difficulty of finding a Nash equilibrium.
Originally, [12] used the Jensen–Shannon divergence as the measure of similarity between data distribution and generated distribution. However, this metric fails to

provide a meaningful value when two distributions do not have substantial overlap, which makes the gradients vanish. In a few years more stable alternatives have been proposed, such as least squares GANs [14] or Wasserstein GANs [15], [16]. Some other training tricks or architecture guidelines have also been released including progressive growing and increasing variation using minibatch standard deviation [17], deep convolutional GANs [13] or even unrolled optimization of the discriminator [18].

4.2 Architecture

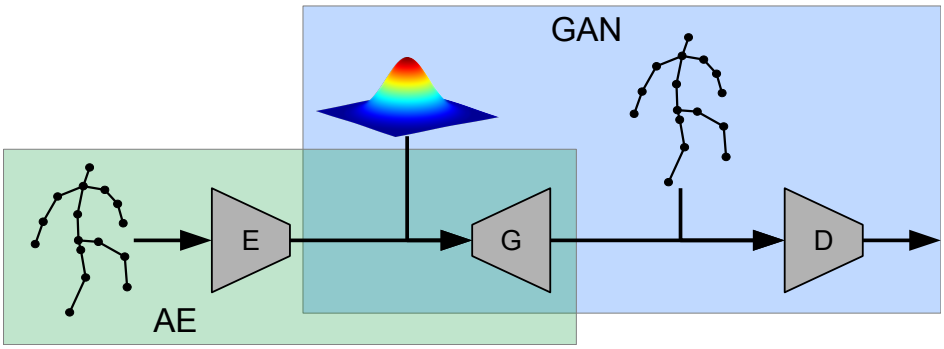


Figure 2: Representation of our deep generative network.

As illustrated in figure 2, our model basically consists of a generative adversarial network augmented with an encoder, forming an autoencoder with the generator playing the role of the decoder. In the following we refer to the autoencoder's decoder as the generator.
Deep Convolutional GANs (DCGANs) being one of the most popular and successful network designs for GANs, we decided to follow it. Therefore, the generator is essentially made of transposed convolutional layers interlaced with batch normalization and ReLUs as activation layers. Analogously, the discriminator is made of convolutional layers interlaced with batch normalization and LeakyReLUs as activation layers. Finally, the encoder has the same architecture as the discriminator, except for the last layer in order to have an output size equal to the latent space dimensionality.
We chose to design our generative model such that it processes fixed-length pose sequences. Indeed [8] showed that the motion cues and temporal consistency of the pose sequence can be better captured by a simple fully connected network operating on fixed-length windows than by a RNN. They believe it happens because intricate human motion has increasing variations possible with increasing time window, which perhaps makes additional information from too far in the time useless or at least difficult to utilize. Therefore, a limited context can effectively capture the useful consistency and motion cues. They additionally observe that poses separated

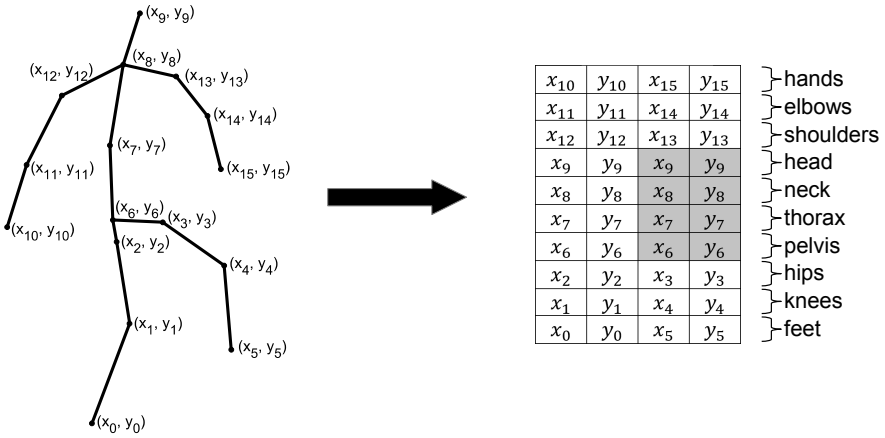
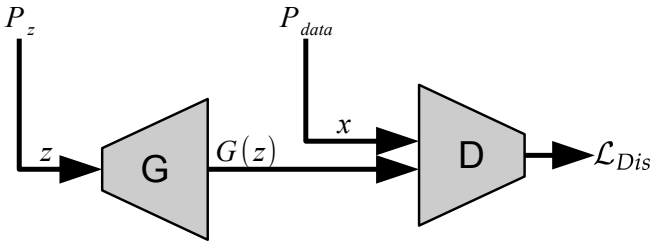


Figure 3: Example of how our sub-networks would internally represent a 16-joints pose. On the right part, the array show how joints coordinates are arranged. The vertical axis of the array corresponds to the hierarchical dimension while the horizontal one is the channeled dimension. Gray cells depict duplicated coordinates. Note that temporal dimension is not represented here for clarity.

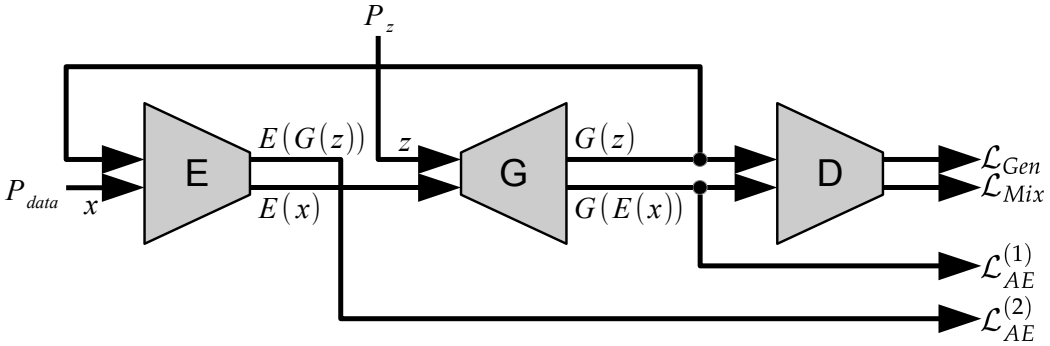
by more than 200ms do not have a noticeable impact on the output of their network.

Data Representation and Convolutions on Pose Sequences DCGAN architecture is designed for and seems particularly adapted for images, due the efficiency of 2D convolutions on images, however we are not processing images, so we had to adapt it to pose sequences.
A 2D pose sequence as input or output of our sub-networks is represented as a 3-dimensional tensor containing the 2D coordinates of each joint at each frame. This representation has two dimensions in space and one in time.
Internally our sub-networks rearrange this representation in order to apply efficient and senseful 2D convolutions. The rearranged representation has two dimensions along which the convolutions are applied, one temporal and one spatially hierarchical following the human skeleton from hands to feet, where each entry holds the coordinates for two joints (i.e. four channels). Dual joints (e.g. feet, knees, etc.) are paired to form an entry, while non-dual ones (e.g. pelvis, thorax, etc.) are duplicated [2] in order to fulfill the four channels at each entry. Figure 3 illustrates how joints coordinates of a single frame are arranged for a 16-joints skeleton topology. The same rearrangement scheme remains applicable to any other topology that has a left-right symmetry.

[2]Both discriminator and encoder duplicate non-dual input joints while the generator produces duplicated non-dual joints in a first step and then outputs the mean over the two versions.



(a) Flow during discriminator's training step.



(b) Flow during generator/encoder's training step.

Figure 4: Flows through our combined AE-GAN model during training.

4.3 Training Procedure

Similarly to a GAN, at each iteration we train our sub-networks within two steps: we optimize the discriminator by minimizing its loss function in a first step while the generator and the encoder are trained both in the adversarial and autoencoder contexts in a second step. Figure 4 illustrates the computational flows through the sub-networks during training for both steps.

4.4 Objectives

Notations In the following, we note our sub-networks G, E and D corresponding respectively to the generator, the encoder and the discriminator. Moreover, P_z and P_{data} denote the latent distribution (also called prior distribution) and the data distribution, where P_z is the standard multivariate normal distribution. Finally we also note for simplicity P_{gen} the generator distribution (i.e. P_z mapped by G) and $P_{\hat{x}}$ the distribution of uniformly sampled points along straight lines between pairs of points sampled from the data distribution P_{data} and the generator distribution P_{gen}, as defined in the improved Wasserstein [16].

Adversarial Objectives For the adversarial objective of our model, we chose the improved Wasserstein loss from [16]. The original WGAN formulation [15] lever-

ages the Wasserstein distance to produce a value function which has better theoretical properties than the original GAN objective. However, WGAN requires the discriminator to be 1-Lipschitz which the authors enforce through weight clipping. [16] demonstrates how weight clipping can lead to undesired behavior and propose to replace it by a gradient penalty which does not suffer from the same problems. Hence, the objective of our discriminator that is depicted in 4a is

$$\mathcal{L}_{Dis} = \mathop{\mathbb{E}}_{\tilde{x}\sim P_{gen}}[D(\tilde{x})] - \mathop{\mathbb{E}}_{x\sim P_{data}}[D(x)] + \lambda_{gp} \mathop{\mathbb{E}}_{\hat{x}\sim P_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]$$

where λ_{gp} if the gradient penalty coefficient, while the generator objective in the adversarial context is

$$\mathcal{L}_{Gen} = -\mathop{\mathbb{E}}_{\tilde{x}\sim P_{gen}}[D(\tilde{x})]$$

Autoencoder Objective As illustrated in figure 4b, the objective $\mathcal{L}_{AE}$ of our autoencoder can be decomposed into two components, $\mathcal{L}_{AE}^{(1)}$ and $\mathcal{L}_{AE}^{(2)}$. The first one corresponds to a traditional autoencoder loss that roughly minimizes differences between the autoencoder's input and output while the latter helps to match the autoencoder's latent space with GAN's prior distribution P_z.
$\mathcal{L}_{AE}^{(1)}$ is constituted of two terms minimizing respectively the joints position and speed errors between the ground truth x and its reconstructed version $\tilde{x} = G(E(x))$. More formally, we used the Mean Per Joint Position Error (MPJPE) to measure joints position errors. The MPJPE is a common measure in human pose estimation and can be formulated as

$$MPJPE(x, \tilde{x}) = \frac{1}{J}\frac{1}{F}\sum_{j=1}^{J}\sum_{f=1}^{F} \left\|x_{j,f} - \tilde{x}_{j,f}\right\|_2$$

where $(\cdot)_{j,f}$ denotes the position of the joint j at frame f and J and F are respectively the number of joints and the number of frames of the sequences.
In analogy with the MPJPE, we defined the Mean Per Joint Speed Error (MPJSE) as

$$MPJSE(x, \tilde{x}) = \frac{1}{J}\frac{1}{F}\sum_{j=1}^{J}\sum_{f=1}^{F-1} \left\| \frac{x_{j,f} - x_{j,f+1}}{\Delta t} - \frac{\tilde{x}_{j,f} - \tilde{x}_{j,f+1}}{\Delta t} \right\|_2$$

where Δt is the time interval between two frames. This secondary term penalizing speed differences acts like a powerful regularizer which helps the convergence in the early iterations and also reduces joints temporal noise of the generated pose sequences.
Hence, $\mathcal{L}_{AE}^{(1)}$ is the weighted sum of the MPJPE and MPJSE:

$$\mathcal{L}_{AE}^{(1)} = \lambda_p \mathop{\mathbb{E}}_{x\sim P_{data}}\Big[MPJPE\Big(x, G\big(E(x)\big)\Big)\Big] + \lambda_s \mathop{\mathbb{E}}_{x\sim P_{data}}\Big[MPJSE\Big(x, G\big(E(x)\big)\Big)\Big]$$

The second component $\mathcal{L}_{AE}^{(2)}$ of our autoencoder's objective focuses on the reconstruction of the latent code z sampled from the prior distribution P_z. It simply minimizes the Mean Squared Error (MSE) between z and its reconstructed version $\tilde{z} = E(G(z))$:

$$\mathcal{L}_{AE}^{(2)} = \lambda_z \mathop{\mathbb{E}}_{z \sim P_z} \Big[MSE\Big(z, E\big(G(z)\big)\Big) \Big]$$

Finally, we have

$$\begin{aligned} \mathcal{L}_{AE} = \mathcal{L}_{AE}^{(1)} + \mathcal{L}_{AE}^{(2)} = & \\ & \lambda_p \mathop{\mathbb{E}}_{x \sim P_{data}} \Big[MPJPE\Big(x, G\big(E(x)\big)\Big) \Big] \\ & + \lambda_s \mathop{\mathbb{E}}_{x \sim P_{data}} \Big[MPJSE\Big(x, G\big(E(x)\big)\Big) \Big] \\ & + \lambda_z \mathop{\mathbb{E}}_{z \sim P_z} \Big[MSE\Big(z, E\big(G(z)\big)\Big) \Big] \end{aligned}$$

where λ_p, λ_s and λ_l are hyperparameters to weigh these losses.

Mixed Objective In addition to the objectives listed above, we further tie the autoencoder to the task of realistic generation: we maximize the discriminator score on the reconstructed pose sequence with respect to both encoder and generator. This additional objective allows to further constrain both encoding and generation processes without penalizing them, and can be expressed as

$$\mathcal{L}_{Mix} = - \mathop{\mathbb{E}}_{x \sim P_{data}} \Big[D\big(G\big(E(x)\big)\big) \Big]$$

Final Objective per Sub-Network In summary, the objectives of the three subnetwors are respectively $\mathcal{L}_{Dis}$ for the discriminator and $\mathcal{L}_{Gen} + \mathcal{L}_{AE} + \mathcal{L}_{Mix}$ for the generator and $\mathcal{L}_{AE} + \mathcal{L}_{Mix}$ for the encoder.

Benefits of the Encoder In our case, the expected benefit of this hybrid architecture is that the encoder can be used to efficiently initialize the latent code when starting the inpainting gradient descent by encoding the input pose sequence to be inpainted. This would make the convergence of the inpainting gradient descent better and faster. Furthermore, as described above, this encoder allows us to partially supervise the task of generation using explicit constraints, such that the MPJPE and MPJSE in our case. Finally, we observed that the addition of the encoder to the generative adversarial network stabilizes the training and thus makes our deep generative network much more stable than a traditional GAN, especially much less sensitive to hyperparameters. We believe that this advantage is due to the secondary

task of autoencoding that explicitly asks to the generator to produce realistic and multi-modal samples.
In summary, the benefits of the encoder are:

- efficient latent code initialization for the inpainting gradient descent;
- partial supervision of the generation;
- decreased sensitivity to hyperparameters;
- easier convergence of the adversarial generative model.

4.5 Spatio-Temporal Variance Regularization

The sharpness of samples generated by a GAN is a strength, however, in our case pose sequence that are noised are visually very unpleasant. A good trade-off between smoothness and sharpness is thus difficult to reach. To solve this issue, we propose to explicitly compute the speed of each joint at each frame of a given input and concatenating it to the input itself, so that the discriminator can use these speeds internally. It encourages the discriminator to reject generated samples that are either temporally too smooth or too sharp.
This idea is conceptually very similar and inspired from Karras et al. [17] that increase generated samples variation by concatenating standard deviation at some point of the discriminator.

4.6 Training for Incomplete Input and Joints Upsampling

We describe here the modifications of the model required for correctly 1) handling pose sequences missing some joints and 2) for upsampling the joints.

Pose Sequence Completion A denoising autoencoder is intentionally trained with noise added to the encoder input while minimizing the loss with respect to the clear input. This has for effect that a denoising autoencoder is robust to noisy input and able to denoise a sample simply by encoding then decoding it.
In the same spirit, when training our model we feed the encoder with x' that is a corrupted version of ground truth x, where some joints are missing at some frames, while minimizing our loss with respect to the clear ground truth x. This has the effect to make our encoder robust to missing joints. In that setting, the objectives expressed according to $\tilde{x} = G(E(x))$ remain the same but with $\tilde{x} = G(E(x'))$ instead.

Joints Upsampling Pushing further the idea described above, we can train our generative model with more detailed pose sequences (i.e. with supplementary joints) but feeding the encoder with sequences where the supplementary joints are missing at all frames for all samples. It forces the network to extract and encode high-level

information of the sequences, from which an realistic sequence with higher details can be generated, instead of simply compressing the input.

5 Pose Sequence Inpainting

Once the model is trained, we can use it for the task of pose sequence inpainting. This process is mainly done through a gradient descent in the latent space. More precisely, it is composed of three steps, that are:

1. Initialization of the latent code from which the gradient descent starts;
2. The gradient descent: iterative optimization of a loss function parameterized with the latent code;
3. Generation of the inpainted pose sequence and post-processing;

5.1 Latent Code Initialization

Most gradient descents are randomly initialized having no better alternative. In our case, the benefit of the encoder to our model is not restrained to stabilizing the adversarial objective, it also provides an efficient latent code initializer: given an input pose sequence x to be inpainted, we simply pass it through the encoder which returns the latent code $z_{init} = E(x)$. Since the encoder is jointly trained with the generator to reconstruct pose sequences, the latent code computed by the encoder is much closer to the optimal solution than a random one.

5.2 Inpainting Gradient Descent Loss Function

The loss function used here is naturally very similar to the one used for training the autoencoder part of our model. In particular we keep both MPJPE and MPJSE terms. Furthermore, to constrain the inpainted pose sequence to be as realistic as possible, we included a regularization term that leverages the discriminator of our model: once trained, it can be seen as an estimator of the quality of a pose sequence. Therefore, we aim at maximizing its output score on the generated pose sequence during the gradient descent. This approach is conceptually very similar to the contextual and prior loss used by Yeh et al. [26].
More formally, let x be the input pose sequence to be inpainted and z the parameter to be optimized through the gradient descent. The loss function is then

$$\mathcal{L}_{inpainting} = \underbrace{\gamma_p \, MPJPE\big(x, G(z|x)\big) + \gamma_s \, MPJSE\big(x, G(z|x)\big)}_{\text{contextual loss}} - \underbrace{\gamma_d \, D\big(G(z|x)\big)}_{\text{prior loss}}$$

where γ_p, γ_s and γ_d are tunable hyperparameters, and $G(z|x)$ is the post-processed version of $G(z)$ given x and is defined in the next section.

5.3 Pose Sequence Generation and Post-Processing

At each step of the gradient descent, we produce a pose sequence $\tilde{x} = G(z)$ from the current latent code z using the generator in order to evaluate the loss function. We additionally use the fact that we are given an input pose sequence x (the one we are inpainting): once $\tilde{x}$ is generated, we translate, scale and rotate it to optimally match x. The optimal translation, scaling and rotation have an analytical solution and the corresponding operations are differentiable. This post-processing method is known as Ordinary Procrustes Analysis [3] (OPA), which is a particular case of Generalized Procrustes Analysis (GPA). Notation: let $OPA(\tilde{x}|x)$ the result of optimally translating, scaling and rotating $\tilde{x}$ to match x, then we write $G(z|x) = OPA\big(G(z)|x\big)$
We observed that this generation post-processing has a low overhead while making the gradient descent convergence several times faster and giving better inpainting results. The final inpainting result is also generated using this method, from the final latent code z^*, i.e. $x^* = G(z^*|x)$.

5.4 Stopping Criterion

The gradient descent implies to define a stopping criterion. Since we do not have a priori knowledge on the input, we have neither knowledge about the optimal score of the loss function. Having no other choice, it is then logical to predefine a number of iterations for the gradient descent. In practice this number of iterations could be adapted: rough but fast inpainting could be done with a low number of iterations while a high one would be time consuming but provide higher quality.

[3] `https://en.wikipedia.org/wiki/Procrustes_analysis`

5.5 Incomplete Input and Upsampling

As defined above, the inpainting essentially refines the pose sequence by reconstructing it through the latent space and constraining it to be generated globally consistent thanks to the GAN. Here we specify what we need to adapt to also complete pose sequences and upsample the joints, beyond model differences explained in section 4.6.
Both completion and upsampling can be expressed in the same manner: we want to inpaint an input pose sequence for which some joints [4] are missing. Latent code initialization is directly done through the encoder as before since it has been correspondingly trained. Concerning the loss function, both MPJPE and MPJSE terms cannot be computed on all joints since some of them are missing in the input. We then only compute these scores on joints that are available, leaving the others only constrained by the generative consistency of the model and by the prior loss. This also applies for the Procrustes Analysis: optimal translation, scaling and rotation are computed only on the available joints.

[4] no specific pattern for completion while a predefined set of joints are additionally missing for upsampling.

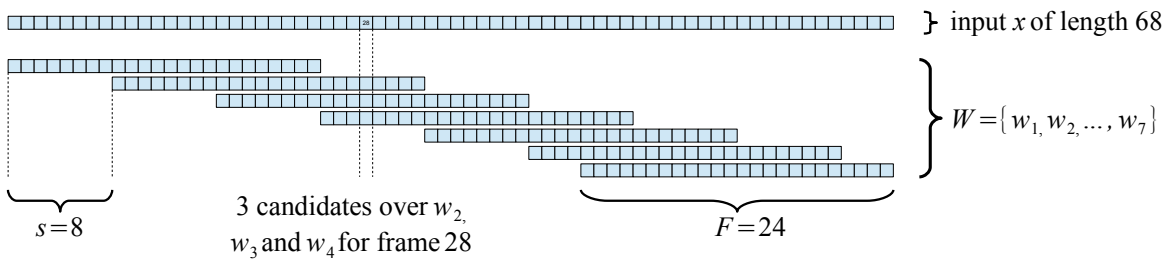


Figure 5: Example of how a set of overlapping fixed-length windows of 24 frames with a shift of 8 is sampled from an input pose sequence of length 68. Each square represents a single frame.

6 Handling Variable Length Sequences

Since our deep generative model processes and produces fixed-length sequences, we describe in this section a simplistic method to handle variable-length sequences upon our fixed-length inpainting method. We assume input sequences length is at least F frames, where F is the length of the sequences supported by our network. Note that shorter sequences could have been interpreted as sequences whose one or more frames are missing and therefore also handled by inpainting.

For an input pose sequence x of length $l \geq F$, we first define a set W of windows (or sub-sequences) of length F sampled at regular frame interval $s \leq F$, hereafter called sub-sequence shift or simply shift. The case $s = F$ corresponds to splitting x into chunks while smaller shifts produce overlapping windows. Figure 5 shows a such overlapping windows sampling procedure. Note the last window that is anchored to the end of the input pose sequence, breaking the regular shift, in order to cover the full sequence.

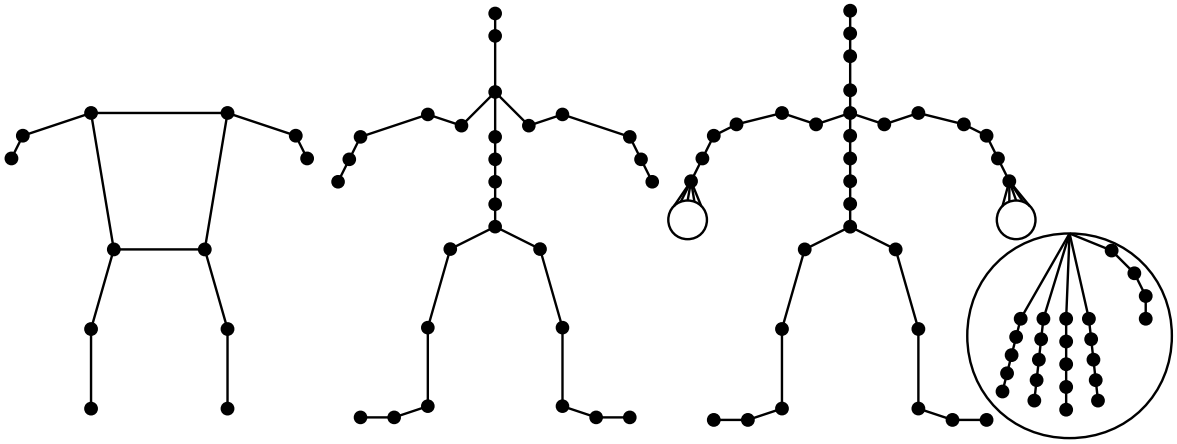
Then, each sub-sequence is independently inpainted using our method described in section 5, giving multiple candidates for the frames being in overlapping regions, as suggested in figure 5.

Finally, we recompose the inpainted pose sequence $\tilde{x}$ of the input pose sequence x by selecting for each input frame x_f the closest inpainted frame to x_f among inpainted frame candidates. Here the metric used to compute distances between candidates and input frame is the MPJPE over the available joints of the input frame x_f.

Remark: the sub-sequence shift is another parameter that can be used to control the output quality at the expense of the time consumed, depending on the application. Indeed, the more inpainted frame candidates we have during selection, the more we expect good results.

7 Evaluation

7.1 Datasets and Preprocessing



(a) (b) (c)

Figure 6: (a) *Alpha Pose* outputs topology (12 joints). (b) Upsampling target topology (28 joints). (c) Proprietary dataset topology (80 joints).

7.1.1 Proprietary Dataset

Initially, a large proprietary dataset was chosen. This dataset is composed of 80-joints based 3D pose sequences acquired by motion capture with markers and whose topology is illustrated in figure 6c. These captures were initially dedicated to film production; therefore, they cover a wide range of activities interpreted by many professional actors ensuring high quality pose sequences. This dataset thus perfectly fills the needs of our deep generative network for learning the human motion distribution without supervision.

Overlapping Windows The first preprocessing step consisted in extracting fixed-length sequences from the ones (variable-length) we have. To do this, we cut out half-overlapping windows from each sequence.

Left-Right Mirroring Most people show a preference for one side of their body over the other. We then considered the human motion to be asymmetric, so does the pose sequences in our dataset. In consequence, for each sequence we added its left-right mirrored version to make our network invariant to laterality.

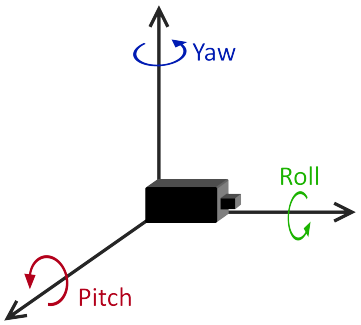


Figure 7: Illustration of the yaw, pitch and roll angles representing the camera orientation.

Joints Subset Since our dataset is composed of highly detailed skeletons (80 joints), we are able to select a subset of the joints as needed to learn a particular human skeleton topology. Note that the topology of most usual datasets in human pose estimation (e.g. MPII [27], COCO [28], MPI-INF-3DHP [29], etc.) are subsets of the 80 joints we have. In practice, we focused on upsampling from 12 to 28 joints, 12 being the number of joints of *Alpha Pose* outputs and 28 seems to be the highest number of joints of public datasets, e.g. MPI-INF-3DHP. As a consequence, we used the joints subset corresponding to MPI-INF-3DHP topology that is depicted in figure 6b.

Randomized Projections Finally, we have 3D joints position but we aim at learning the two dimensional human motion distribution, we then have to project these data. We made it a benefit by taking multiple randomized projections from 3D to 2D of our dataset using various camera locations in 3D space.
In order to have sufficient variance and meaningful projected data, we used a randomized procedure to define a projection from a camera setting for each 3D pose sequence. A camera setting here can be summarized as the camera orientation, i.e. yaw, pitch and roll angles (see figure 7), and its position. Both pitch and roll angles have been uniformly sampled, so does the camera height. Yaw angle and camera horizontal position have been computed such that the camera points toward the mean joints position of the sequence considered and that the sequence entirely lies in the square of size 2 centered at the origin, producing a normalized 2D pose sequence.

In summary, we have at our disposal about 12 million projected 2D pose sequences whose resulting numerical properties are the following:

- The mean position over joints and frames of each sample is $(0, 0)$.
- The position of all joints for each sample lie in $[-1, 1]^2$.

7.1.2 MPI-INF-3DHP

Since the proprietary dataset has only 3D pose sequences, we required another dataset to fully evaluate the proposed human pose estimation method, from image sequences to 28 joints 2D pose sequences. We chose MPI-INF-3DHP [29] for that purpose.
MPI-INF-3DHP dataset has been captured in a multi-camera studio with ground truth from commercial marker-less motion capture. There are 8 actors performing 8 activity sets each ranging from walking and sitting to complex exercise poses and dynamic actions. Each actor features 2 sets of clothing split across the activity sets. A wide range of viewpoints is covered by 14 cameras at different angles and heights.
This dataset is splitted into two parts, a trainset and a testset. Unfortunately, the testset only contains 17-joints pose sequences hence cannot be used for assessing our model that upsamples poses to 28 joints. In consequence, we used a subset of the trainset to evaluate our model.
3 of the 14 cameras are top views of the scene which gives strange 2D projected pose which does not correspond to anything that *Alpha Pose* has learned. We then decided to discard these cameras for both training and testing.
We defined our testset as the set containing the sequences of first actor; however, we discarded all subsequences for which the actor is going out of field. More precisely, we cut each of these subsequences from the moment the actor is not entirely visible anymore until he is entirely visible again. Note that we kept subsequences where the actor is only partially out of field, e.g. when he comes close the the camera and that its legs are out of field. In summary, our testset is made of 70 sequences from 87 frames (or $\sim$ 3.6s) for the shorter to 12'400 frames (or $\sim$ 8m 37s) for the longer, for a total of 196'178 frames.
Our trainset is then composed of the remaining 7 actors. In order to have the most similar numerical and statistical properties for both proprietary and MPI-INF-3DHP trainsets, we applied the same procedures as described in section 7.1.1, i.e. half-overlapping windowing, left-right mirroring and randomized projections. We thus used MPI-INF-3DHP 3D annotations to obtain a trainset made of about 700'000 fixed-length 2D pose sequences.

7.2 Implementation Details

7.2.1 Model

The architecture of each sub-network is detailed in table 1. Each one contains a "rearrangement" layer corresponding to the change in data representation making possible convolutions on pose sequences, as described in section 4.2. Note that these layers have no learnable parameter and are shown here for input and output shape consistency and comprehension. The "speed concat." layer refers to spatio-temporal variance regularization proposed in section 4.5.

layer	kernel	stride	BN	Activation	input shape	output shape	params
conv	1 x 1	1 x 1	yes	ReLU	128 x 1 x 1	256 x 1 x 1	33′280
conv	3 x 3	1 x 1	yes	ReLU	256 x 1 x 1	256 x 3 x 3	590′336
conv	4 x 3	2 x 1	yes	ReLU	256 x 3 x 3	128 x 6 x 5	393′472
conv	4 x 3	2 x 2	yes	ReLU	128 x 6 x 5	64 x 12 x 9	98′432
conv	4 x 4	2 x 2	yes	ReLU	64 x 12 x 9	32 x 24 x 18	32′832
conv	1 x 1	1 x 1	no	Tanh	32 x 24 x 18	4 x 24 x 18	128
rearrangement	-	-	-	-	4 x 24 x 18	2 x 24 x 28	-
Total							1′148′480

(a) Generator

layer	kernel	stride	BN	Activation	input shape	output shape	params
rearrangement	-	-	-	-	2 x 24 x 28	4 x 24 x 18	-
speed concat.	-	-	-	-	4 x 24 x 28	8 x 24 x 18	-
conv	1 x 1	1 x 1	no	LReLU	8 x 24 x 18	32 x 24 x 18	256
conv	4 x 4	2 x 2	yes	LReLU	32 x 24 x 18	64 x 12 x 9	32′896
conv	4 x 3	2 x 2	yes	LReLU	64 x 12 x 9	128 x 6 x 5	98′560
conv	4 x 3	2 x 1	yes	LReLU	128 x 6 x 5	256 x 3 x 3	393′728
conv	3 x 3	1 x 1	no	-	256 x 3 x 3	256 x 1 x 1	589′824
linear	-	-	no	sigmoid	256 x 1 x 1	1 x 1 x 1	257
Total							1′115′521

(b) Discriminator

layer	kernel	stride	BN	Activation	input shape	output shape	params
rearrangement	-	-	-	-	3 x 24 x 28	6 x 24 x 18	-
conv	1 x 1	1 x 1	no	LReLU	6 x 24 x 18	32 x 24 x 18	192
conv	4 x 4	2 x 2	yes	LReLU	32 x 24 x 18	64 x 12 x 9	32′896
conv	4 x 3	2 x 2	yes	LReLU	64 x 12 x 9	128 x 6 x 5	98′560
conv	4 x 3	2 x 1	yes	LReLU	128 x 6 x 5	256 x 3 x 3	393′728
conv	3 x 3	1 x 1	no	-	256 x 3 x 3	256 x 1 x 1	589′824
linear	-	-	no	sigmoid	256 x 1 x 1	128 x 1 x 1	32′896
Total							1′148′096

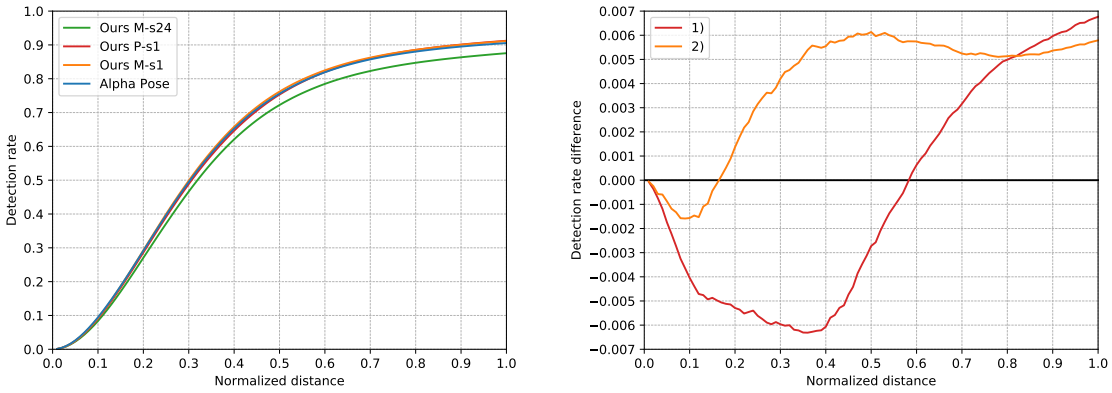
(c) Encoder

Table 1: Layers of each sub-network.

The shape of a pose sequence (output shape of the generator and input shape of the discriminator and the encoder) is $C \times F \times J$ where $C \in \{2, 3\}$ is the number of coordinates, $F = 24$ is sequence length and $J = 28$ the number of joints. Unlike the generator and the discriminator, the encoder processes pose sequences with a third coordinate which allows for indicating joints position that are missing (0 or 1), both at training and testing time. In other words, this third coordinate on its own acts like a mask of the pose sequence. The input size of the generator and the output size of the encoder correspond to the dimensionality of the latent space L, set to $L = 128$ for all experiments.

7.2.2 Training

We trained our model for 50 million iterations with a minibatch size of 256 using one Adam optimizer [1] for each sub-network, each with $\alpha = 0.0001$, $\beta_1 = 0.5$,



(a) (b)

Figure 8: Left: PCKh curves for $\alpha \in [0,1]$ over 12 joints. Right: PCKh differences w.r.t. Alpha Pose; 1) $PCKh(\text{Ours P-s1}) - PCKh(\text{Alpha Pose})$; 2) $PCKh(\text{Ours M-s1}) - PCKh(\text{Alpha Pose})$.

$\beta_2 = 0.999$ and $\epsilon = 10^-8$.
We set the Wasserstein gradient penality weight λ_{gp} to 10 as proposed in [16], and our loss weights λ_p, λ_s and λ_z to 200, 100 and 2 but note that the choice of our loss weights is highly dependent on the scale of the 2D pose sequences, or in other words, their relative magnitudes are more relevant here than their respective values.

7.2.3 Inpainting

For the gradient descent of our proposed inpainting method, we again used an Adam optimizer, with $\alpha = 1.0$, $\beta_1 = 0.8$, $\beta_2 = 0.999$ and $\epsilon = 10^-8$.
The weights of the inpainting objective function have been chosen as follow: $\gamma_p = 10.0, \gamma_s = 5.0, \gamma_d = 0.5$. Finally, we stop gradient descent after 200 steps.

7.3 Evaluation Metrics

We used the standard Percentage of Correct Keypoints (PCK) metric. An estimated keypoint (joint position) is considered correct if its distance from ground truth is less than a fraction α of a segment length of the ground truth. In particular, the PCKh uses the ground truth head size to normalize distances. This metric is denoted PCKh@α. In addition to the PCKh, we also rely on the Area Under the Curve (AUC) that measures the area under the PCKh@α curve for α in a given range.

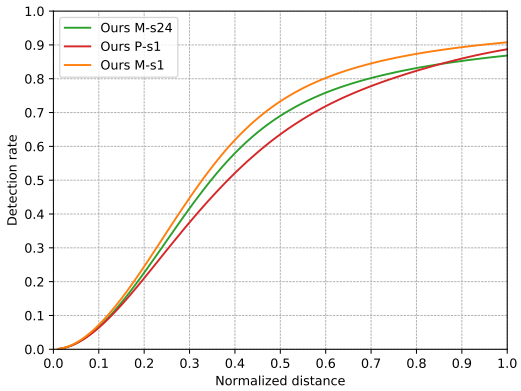


Figure 9: PCKh curves for $\alpha \in [0, 1]$ over all 28 joints.

7.4 Experimental Results

We either trained our deep generative network with MPI-INF-3DHP or our proprietary trainset as defined in section 7.1.2. In a second phase, we ran *Alpha Pose* on the MPI-INF-3DHP testset also defined in section 7.1.2 and obtained first 2D pose sequences estimation. These estimations allow to compute the scores that *Alpha Pose* reach but also to apply the proposed inpainting method.
We denote our experiments as "Ours {M, P}-s*n*" where M and P letters define respectively MPI-INF-3DHP dataset and proprietary dataset used for training while *n* define the sub-sequence shift used during inpainting.
Table 2 provides PCKh@0.1, PCKh@0.5 and PCKh@1.0 and AUC scores of *Alpha Pose* estimations and inpainted ones. This table focus on the 12 joints handled by *Alpha Pose* in order to rightfully compare our predictions to *Alpha Pose* outputs. Table 3 gives the same scores for the 16 joints upsampled and the overall average.
Figure 8a shows the PCKh curves for 12 joints. Since PCKh curves for "Ours P-s1", "Ours M-s1" and *Alpha Pose* are almost superimposed, we also provide the differences between our methods and *Alpha Pose* (we subtract *Alpha Pose*'s PCKh to ours) in figure 8b. Figure 9 also shows PCKh curves but computed over all 28 joints, thus excluding *Alpha Pose* . Finally figures 10 and 11 gives visual comparison between ground truth, *Alpha Pose* outputs and inpainted poses.

Method	Ankles	Knees	Hips	Shoulders	Elbows	Wrists	Total 12J
Alpha Pose	**0.094**	0.076	0.023	0.076	**0.112**	**0.185**	**0.094**
Ours M-s1	0.091	**0.077**	0.024	0.078	0.111	0.175	0.093
Ours M-s24	0.081	0.072	**0.025**	0.074	0.102	0.148	0.084
Ours P-s1	0.088	0.073	0.018	**0.084**	0.110	0.169	0.090

(a) PCKh@0.1

Method	Ankles	Knees	Hips	Shoulders	Elbows	Wrists	Total 12J
Alpha Pose	0.737	0.768	0.605	0.892	0.786	0.746	0.756
Ours M-s1	**0.739**	**0.782**	**0.621**	**0.900**	0.783	**0.748**	**0.762**
Ours M-s24	0.691	0.741	0.603	0.847	0.743	0.709	0.722
Ours P-s1	0.736	0.773	0.577	0.899	**0.787**	0.747	0.753

(b) PCKh@0.5

Method	Ankles	Knees	Hips	Shoulders	Elbows	Wrists	Total 12J
Alpha Pose	0.853	0.901	0.938	0.959	0.919	0.864	0.906
Ours M-s1	**0.861**	**0.907**	**0.946**	**0.967**	0.917	0.869	0.911
Ours M-s24	0.824	0.870	0.915	0.924	0.881	0.839	0.876
Ours P-s1	0.860	0.906	0.944	**0.967**	**0.924**	**0.873**	**0.912**

(c) PCKh@1.0

Method	Ankles	Knees	Hips	Shoulders	Elbows	Wrists	Total 12J
Alpha Pose	**0.603**	0.618	0.531	0.698	0.653	**0.640**	0.624
Ours M-s1	**0.603**	**0.626**	**0.540**	0.706	0.651	**0.640**	**0.628**
Ours M-s24	0.567	0.596	0.524	0.667	0.619	0.605	0.596
Ours P-s1	0.601	0.620	0.518	**0.707**	**0.654**	0.639	0.623

(d) AUC from 0.0 to 1.0

Table 2: PCKh@$\{0.1, 0.5, 1.0\}$ and AUC scores on 12 joints.

7.5 Discussion

12 joints human pose estimation As shown in table 2 and figure 8a, our method "M-s1" slightly improves the 12 joints pose estimations made by *Alpha Pose* for $\alpha > 0.16$, while producing competitive 16 supplementary joints for which the detection rates are close to the ones for the 12 joints (-2%, -2.9% and -0.3% for normalized distance 0.1, 0.5 and 1). For $\alpha < 0.16$, the detection rates of both *Alpha Pose* and "M-s1" are relatively low (under 25%) and the difference between them is very small (less than 0.2%). Figure 10 provides an example where *Alpha Pose* fails at estimating the movement of the right forearm while this movement has been recovered by inpainting.
"P-s1" doesn't improve 12 joints estimation under normalized distance of 0.57 which is not negligible but doesn't give poor results (at most 0.7% below *Alpha Pose*).

Method	Toes	Feet	Spine	Head	Clavicles	Hands	Total 28J
Ours M-s1	0.024	0.031	**0.059**	**0.046**	**0.076**	**0.096**	**0.072**
Ours M-s24	0.023	0.030	0.057	0.041	0.067	0.085	0.066
Ours P-s1	**0.031**	**0.050**	0.034	0.045	0.034	0.089	0.064

(a) PCKh@0.1

Method	Toes	Feet	Spine	Head	Clavicles	Hands	Total 28J
Ours M-s1	0.415	0.492	**0.847**	**0.652**	**0.916**	**0.678**	**0.733**
Ours M-s24	0.389	0.463	0.798	0.593	0.857	0.632	0.690
Ours P-s1	**0.453**	**0.578**	0.451	0.570	0.755	0.670	0.636

(b) PCKh@0.5

Method	Toes	Feet	Spine	Head	Clavicles	Hands	Total 28J
Ours M-s1	**0.780**	0.822	**0.970**	**0.909**	**0.978**	0.840	**0.908**
Ours M-s24	0.739	0.782	0.931	0.854	0.934	0.804	0.869
Ours P-s1	0.768	**0.825**	0.890	0.870	0.974	**0.841**	0.887

(c) PCKh@1.0

Method	Toes	Feet	Spine	Head	Clavicles	Hands	Total 28J
Ours M-s1	0.399	0.452	**0.663**	**0.554**	**0.710**	**0.571**	**0.603**
Ours M-s24	0.376	0.427	0.629	0.510	0.667	0.536	0.570
Ours P-s1	**0.416**	**0.500**	0.442	0.505	0.610	0.564	0.547

(d) AUC from 0.0 to 1.0

Table 3: PCKh@$\{0.1, 0.5, 1.0\}$ and AUC scores on supplementary (upsampled) joints and average over all 28 joints.

Inpainting sub-sequence shift Tables 2 and 3 and figures 8 and 9 all tells us that small sub-sequence shift (i.e. "M-s1") outperforms a large one (i.e. "M-s24") in average. However, as suggested in figure 11, a large shift is better when the subject is partially out of field.
We believe it happens because of the selection process among candidates. When the shift is equal to the window size (i.e. 24) there is no overlap and therefore only one candidate per frame (or no selection at all). On the opposite, the smaller the shift, the higher the candidates. Since we select the candidate with the lowest mean position error over the available joints, it is not surprising that this selection process is not appropriate for missing joints estimation, particularly if too many joints are missing or when they are not homogeneously distributed. This is why a large window shift yields better results in situations such that figure 11.

Inpainting limitations While being able to predict plausible upsampled 2D pose, our inpainting method using only human motion priors has some limitations. One of them is depicted in figure 10: the orientation of the head is not correctly predicted,

although the right forearm is almost correctly recovered. This kind of limitations seems to appear when the motion of a body part is not sufficiently correlated to the rest of the pose and in particular it often concerns the head. In these cases, a plausible prediction can be far from the correct one.

Specific training The figure 11 shows the difficulty to correctly reconstruct entirely missing limbs. Indeed, the reconstruction of the legs when they are entirely out of the camera field is better than nothing since the global orientation and their length are plus ou moins correct. However, the estimated legs are far from being plausible. Nevertheless, our generative model could be specifically trained for a such complex task. We trained our generative model to be robust to missing joints and partial input poses (encoding of 12-joints poses while generating 28-joints poses). Similarly, we could have trained the same architecture but dedicated to the task of reconstructing the lower body by only feeding the encoder with the upper human body. More generally, this kind of specific training could be adapted to any subset of the human body joints depending on the targeted application.

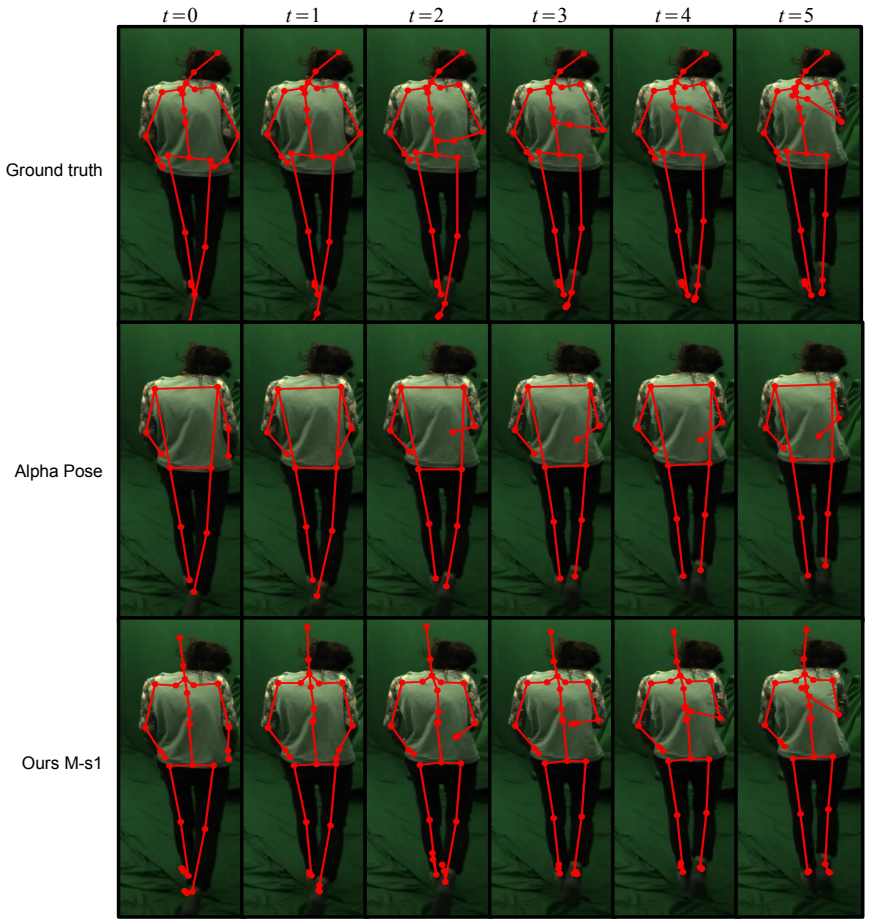


Figure 10: Example of sequence where Alpha Pose fails at estimating the movement of an occluded limb (here the right hand, down to top) but inpainting succeeds in reconstructing it. It also depicts limitation of upsampling: the head reconstructed is plausible but incorrect given the ground truth.

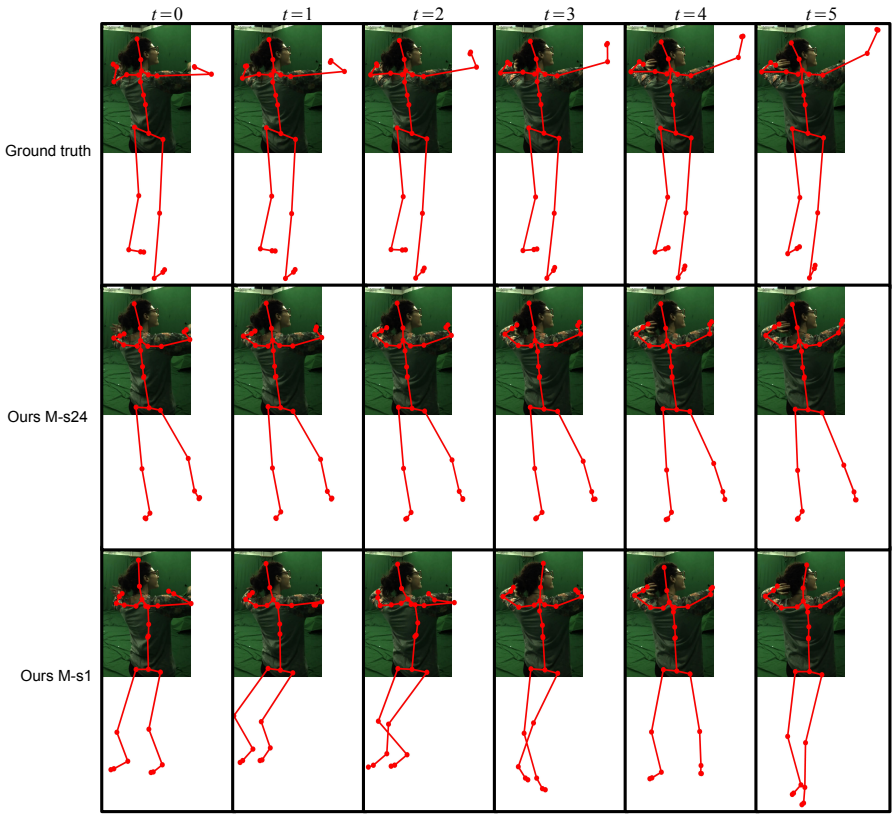


Figure 11: Example of inpainting beyond camera field of view: it illustrates that larger inpainting sub-sequence shift allows to better capture the motion cues.

8 Conclusion

In summary, we presented a novel human poses inpainting method relying on an hybrid adversarial generative model that improves one of the most accurate 2D human pose estimator by completing, refining and upsampling estimated poses, using human motion priors.
Our framework takes a 12-joints 2D pose sequence as input and produces a valuable 28-joints 2D pose sequence by inpainting the input. The generative model proposed consists of the fusion of a deep convolutional generative adversarial network and an autoencoder. An input pose sequence is inpainted by optimizing its latent representation computed by the encoder using a contextual loss to match the input and a prior loss to ensure a plausible generated output.
We also proposed a simplistic method to handle variable-length input sequence on top of our inpainting method. It essentially splits the input into fixed-length overlapping windows and then aggregates inpainted windows into a single output of the same length as the input.
The results we obtained are very encouraging and leave a lot of interesting future works. First of all, a such method could further explicitely enforce the generated pose sequences to be realistic using a geometric or biomechanical model of human body movements. Then, another additional stage to the 3D human pose estimation pipeline could be used: the adaptation to the 3D would potentially allow to complete (if not already done) and refine 3D pose estimations after depth regression. Finally, we could also try to directly inpaint estimated 2D poses to infer the depth using a generative model that would have learned the 3D human motion distribution.
Future work should also investigate richer temporal analysis by either using a different network architecture handling variable-length pose sequences (e.g. based on C-RNN-GAN [19] or fully convolutional networks) or training a deep network dedicated to inpainted windows composition.
Finally, the method proposed being very flexible and easily adaptable, it could be used in a wide variety of applications such that human motion edition or character animation and many others,

References

[1] D. Kingma, J. Lei Ba, *Adam: A Method for Stochastic Optimization*, In ICLR, 2015

[2] A. Toshev, C. Szegedy, *DeepPose: Human pose estimation via deep neural networks*, In CVPR, 2014

[3] A. Newell, K. Yang, J. Deng *Stacked Hourglass Networks for Human Pose Estimation*, In ECCV, 2016

[4] H. Fang, S. Xie, Y. Tai, C. Lu, *RMPE: Regional Multi-Person Pose Estimation*, In ICCV, 2017

[5] Y. Xiu, J. Li, H. Wang, Y. Fang, C. Lu, *Pose Flow: Efficient Online Pose Tracking*, In BMCV, 2018

[6] B. Tekin, I. Katircioglu, M. Salzmann, V. Lepetit, P. Fua, *Structured Prediction of 3D Human Pose with Deep Neural Networks*, In BMVC, 2016

[7] X. Zhou, Q. Huang, X. Sun, X. Xue, Y. Wei, *Towards 3D Human Pose Estimation in the Wild: a Weakly-supervised Approach*, In ICCV, 2017

[8] R. Dabral, A. Mundhada, U. Kusupati, S. Afaque, A. Sharma, A. Jain, *Learning 3D Human Pose from Structure and Motion*, In ECCV, 2018

[9] M. Lin, L. Lin, X. Liang, K. Wang, H. Cheng, *Recurrent 3D Pose Sequence Machines*, In CVPR, 2017

[10] Y. Chen, C. Shen, X. Wei, L. Liu, J. Yang, *Adversarial PoseNet: A Structure-aware Convolutional Network for Human Pose Estimation*, In ICCV, 2017

[11] W. Yang, w. Ouyang, X. Wang, J. Ren. H. Li, X. Wang, *3D Human Pose Estimation in the Wild by Adversarial Learning*, In CVPR, 2018

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, *Generative Adversarial Nets*, In NIPS, 2014

[13] A. Radford, L. Metz, S. Chintala, *Unsupervised Representation Learning with Deep Convolutional Generative Networks*, In ICLR, 2016

[14] X. Mao, Q. Li, H. Xie, R. Lau, Z. Wang, S. Smolley, *Least Squares Generative Adversarial Networks*, In ICCV, 2017

[15] M. Arjovsky, S. Chintala, L. Bottou, *Wasserstein GAN*, arXiv:1701.07875, 2017

[16] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. Courville, *Improved Training of Wasserstein GANs*, In NIPS, 2017

[17] T. Karras, T. Aila, S. Laine, J. Lehtinen, *Progressive Growing of GANs for Improved Quality, Stability, and Variation*, In ICLR, 2018

[18] L. Metz, B. Poole, D. Pfau, J. Sohl-Dickstein, *Unrolled Generative Adversarial Networks*, In ICLR, 2017

[19] O. Mogren, *C-RNN-GAN: Continuous recurrent neural networks with adversarial training*, arXiv:1611.09904, 2016

[20] P. Vincent, H. Larochelle, Y. Bengio, P. Manzagol, *Extracting and Composing Robust Features with Denoising Autoencoders*, In ICML, 2008

[21] D. Kingma, M. Welling, *Auto-Encoding Variational Bayes*, In ICLR, 2014

[22] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, B. Frey, *Adversarial Autoencoders*, In ICLR, 2016

[23] A. Boesen, S. Kaae, H. Larochelle, O. Winther, *Autoencoding beyond pixels using a learned similarity metric*, In PMLR, 2016

[24] J. Bao, D. Chen, F. Wen, H. Li, G. Hua, *CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training*, In ICCV, 2017

[25] T. Sainburg, M. Thielk, B. Theilman, B. Migliori, T. Gentner, *Generative Adversarial Interpolative Autoencoding: Adversarial Training on Latent Space Interpolations Encourages Convex Latent Distributions*, arXiv:1807.06650, 2018

[26] R. Yeh, C. Chen, T. Yian Lim, A. Schwing, M. Hasegawa-Johnson, M. Do, *Semantic Image Inpainting with Deep Generative Models*, In CVPR, 2017

[27] M. Andriluka, L. Pishchulin, P. Gehler, B. Schiele, *2D Human Pose Estimation: New Benchmark and State of the Art Analysis*, In CVPR, 2014

[28] T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. Zitnick, P. Dollar, *Microsoft COCO: Common Objects in Context*, In ECCV, 2014

[29] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, C. Theobalt, *Monocular 3D Human Pose Estimation In The Wild Using Improved CNN Supervision*, In 3DV, 2017

[30] M. Andriluka, U. Iqba, E. Insafutdinov, L. Pishchulin, A. Milan, J. Gall, B. Schiele, *PoseTrack: A Benchmark for Human Pose Estimation and Tracking*, In CVPR, 2018