MoGlow: Probabilistic and controllable motion synthesis using normalising flows

Gustav Eje Henter[*] Simon Alexanderson[*] Jonas Beskow
Division of Speech, Music and Hearing, KTH Royal Institute of Technology, Stockholm, Sweden
{ghe,simonal,beskow}@kth.se

Abstract

Data-driven modelling and synthesis of motion data is an active research area with applications that include animation and games. This paper introduces a new class of probabilistic, generative motion-data models based on normalising flows, specifically Glow. Models of this kind can describe highly complex distributions (unlike many classical approaches like GMMs) yet can be trained stably and efficiently using standard maximum likelihood (unlike GANs). Several model variants are described: unconditional fixed-length sequence models, conditional (i.e., controllable) fixed-length sequence models, and finally conditional, variable-length sequence models. The last type uses LSTMs to enable arbitrarily long time-dependencies and is, importantly, causal, meaning it only depends on control and pose information from current and previous timesteps. This makes it suitable for generating controllable motion in real-time applications. Every model type can in principle be applied to any motion since they do not make restrictive assumptions such as the motion being cyclic in nature. Experiments on a motion-capture dataset of human locomotion confirm that motion (sequences of 3D joint coordinates) sampled randomly from the new methods is judged as convincingly natural by human observers.

1. Introduction

A recurring problem in computer animation is how to generate convincing motion conditioned on external parameters. Consider, for example, a computer-game character that is controlled in real time from a gamepad. Based on the control signals, the character should not only move realistically in different directions, but also change the style of locomotion between, e.g., walking and running as well as perform various actions such as jumping or dodging. This type of problem is traditionally solved by splicing together "canned" motion segments recorded using motion capture. The style of motion produced by such systems may look realistic at first sight, but humans observers rapidly catch on to the highly deterministic and repetitive nature of motion-capture playback, makes it very hard and costly to build truly believable interactive motion with this methodology.

The advent of deep learning and the growing availability of large motion-capture databases have increased the interest in using generative models that produce motion based on statistical models, instead of direct playback of motion clips. In general terms, real-time interactive systems require models with the ability to generate complex and naturalistic motion given only a *weak control signal* (e.g., walk in direction X at a pace of Y m/s). It is important to note that there usually are many possible motion realisations that satisfy any given control signal – the limbs of a real person who is asked to walk the same path twice, at the same speed, would always follow different trajectories. Deterministic models of motion, which return a single predicted motion such as the estimated average pose for each time frame, suffer from regression to the mean pose and produce artefacts like foot sliding in the case of gait. Taken together, we are led to conclude that for motion generated from the model to be perceived as realistic, it *cannot* be completely deterministic, but the model should instead generate *different* motions upon each subsequent invocation, given the same control signal. In other words, a stochastic model is required.

This paper introduces normalising flows [6, 7, 8, 26] for generating motion-data sequences, both of a fixed length and in a causal autoregressive model. This new modelling paradigm has the following principal advantages:

1. It is *probabilistic*, meaning that it does not just describe one plausible motion, but endeavours to describe *all* possible motions, and how likely each possibility is. This avoids the "mean collapse" issue of many models that are trained on deterministic loss functions such as mean squared error (MSE). In the absence of conclusive control-signal input, a well-trained probabilistic model will return plausible candidate-motion samples.
2. It uses *implicit* models to parameterise distributions. Consequently, it is fast to sample from without assum-

[*] Equal contribution.
This research was supported by the Swedish Research Council, proj. 2018-05409, and the Swedish Foundation for Strategic Research, contract no. RIT15-0107.

ing that observations follow restrictive, low-degree-of-freedom parametric families such as Gaussians and Gaussian mixture models (GMMs).
3. It allows exact and tractable probability computation, unlike variational autoencoders (VAEs) [27, 41], and can be *trained to maximise likelihood directly*, unlike generative adversarial networks (GANs) [11, 10].
4. It is *general* – that is, it does not rely on restrictive, situational assumptions such as the motion being periodic or quasi-periodic (in contrast to, e.g., [17]).
5. It can be adapted to generate sequences autoregressively and sequentially, rather than all at once, and allows causal, *zero-latency control* of the output motion.
6. It is capable of generating *high-quality motion* as judged by human observers.

To the best of our knowledge, these are the first motion models based on normalising flows. The most closely related methods outside of motion are WaveGlow [38] and FloWaveNet [24] for audio waveforms, and the very recent VideoFlow [29], all of which use the ideas from Glow [26] to model time-dependent data. We extend these models in several novel directions: Unlike the audio models, our architecture is autoregressive, avoiding costly dilated convolutions; unlike the video model, our architecture permits output control. Unlike all prior flow-based sequence models, we add a hidden state to enable long-range memory, which significantly improves the model. We also present a dropout scheme that enhances the consistency of the motion control and the realism of long motion-sequence samples.

The remainder of this paper is organised as follows: Sec. 2 describes related work in sequence modelling and motion synthesis. Sec. 3 then describes Glow and how we adapt it to model fixed-length motion sequences. Our proposed autoregressive controllable motion model is detailed in Sec. 4. Sec. 5 reports on our experiments while Sec. 6 concludes. A video presentation of our work with generated motion examples can be found at youtu.be/lYhJnDBWyeo.

2. Background and related work

This section reviews recent developments in deep-learning-based generative models and introduce prior art in the domain of motion generation from motion-capture data.

2.1. Probabilistic generative sequence models

Probabilistic sequence models have a long history beginning with linear autoregressive models. These are simple models where inference, parameter estimation, and sampling are fast and easy, at the expense of expressivity. Model flexibility improved with the introduction hidden Markov models [39] and Kalman filters, which still allow tractable exact inference. All of these paradigms have been extensively used in generative sequence models. Unfortunately, these models are still too inflexible to describe complex signals such as motion and speech, as can be seen by the poor quality of random samples from these models (cf. [48]).

Deep learning has enabled more advanced autoregressive models of continuous-valued data, such as [12, 53, 47], where outputs remain explicitly defined as Gaussians or mixture distributions, for tractable inference. These models are however still not sufficiently expressive for many applications. Many of the strongest deep and probabilistic autoregressive models currently available, such as [50, 43, 22], model low-dimensional vectors ($\mathbb{R}^3$ or less) in time or space, and it is not clear how they may be scaled up to data such as motion-data sequences with 50 or more dimensions.

Recent research into deep generative models for complex data has also explored two alternative paths: One is variational autoencoders [27, 41], which (approximately) optimise a variational lower bound on model likelihood, while simultaneously learning to perform approximate inference. The approximations create a notable gap between the true maximum likelihood and that achieved by VAEs [5]. The other is generative adversarial networks [11], which describe distributions that are easy to sample from but do not allow inference, and instead are trained by means of a game against an adversary. GANs have produced some very impressive results in applications such as image generation [1], but their optimisation is fraught with difficulty [33, 32].

It has been found that successfully-trained GANs often produce higher-quality output than VAEs in applications such as image generation. This has been hypothesised to be a consequence of GANs optimising a different objective than maximum likelihood [20, 46] but this is contradicted by other evidence [10, Sec. 3.2.5]. We instead believe the GAN advantage is due to the implicit nature of GAN generators, meaning that output is produced via a deep, nonlinear transformation of samples from a simple latent-space distribution [34]. In principle, VAEs have a partially-implicit generator structure, but due to significant training issues (sometimes called "posterior collapse"), VAEs with strong decoders yield models where latent variables have little impact on output samples [21, 42]. This largely nullifies the benefits of the implicit generator structure.

This article considers a less well-known methodology called normalising flows [6, 7, 8]. We believe these combine the best of both worlds, being implicit models that combine a basis in likelihood and efficient inference like VAEs (but without requiring approximations) with purely implicit generator structures (like GANs). A recent improvement on normalising flows called Glow [26] grabbed attention by producing perhaps the most realistic-looking image samples thus far from a model trained using maximum likelihood. Sequence-modelling applications of these methods are only just emerging, e.g., [38, 24, 29]. Our paper presents one of the first Glow-based sequence models, and the first to our knowledge to combine autoregression and control, and to integrate long-memory via a hidden state, as well as to improve control precision via input-side dropout.

2.2. Data-driven motion synthesis

While early motion synthesis used hand-coded rules and animations, a strong trend has been towards using data-driven methods on motion-capture data. Methods based on *motion graphs* [28] act by concatenating short segments from the training database into novel configurations. While such methods allow control and can be used with relatively small data sets, they do not generalise well and can only produce motion already present in the data. They also suffer from increased latency, as synthesis needs to await a concatenation point before responding to the control signal.

Statistical, generative motion models include Gaussian process latent variable models [14, 30], linear dynamic models [3], convolutional [19], and recurrent neural networks [15, 9]. While many studies have applied neural nets to forecast future poses for a given pose sequence [2, 9, 37], we are particularly interested in *controllable* motion generation. Important for such domains is how strong a predictor the input signal is for the output motion. Lip motion is for example highly predictable from speech and has been successfully modelled with deterministic methods [45, 23]. However, for weaker control, such methods generally fail to disambiguate natural variation and collapse to a mean pose. Various techniques have been proposed to disentangle the variation. For locomotion synthesis [18, 17, 37] the periodic nature of the motion has been exploited, or the disambiguation problem has been divided into simpler sub-tasks (first generating foot steps and then body poses).

Deep probabilistic generative models are still a rarity in motion generation, but VAEs in various forms have been applied to model human locomotion along a given path [15] and to generate head motion from speech [13], while GANs have been applied to generate video of face motion [51]. Our method represents the first probabilistic motion model based on normalising flows, providing advantages in terms of control, responsiveness, stability, and generality.

3. Glow for fixed-length sequences

In this section we describe the mathematical basis of normalising flows, particularly Glow, and how they can be adapted to generate fixed-length motion sequences.

3.1. Preliminary notation

In the following, vector-valued quantities and sequences thereof are denoted with bold font; upper case is used for random variables and matrices, and lower case for deterministic quantities or specific outcomes of the random variables. For example, $\boldsymbol{X}$ typically represents randomly-distributed motion with $\boldsymbol{x} \in \mathbb{R}^{T \times D}$ being an outcome of the same. Non-bold capital letters generally denote indexing ranges, with matching lower-case letters representing the indices themselves, e.g., $t \in \{1, \ldots, T\}$. Indices into sequences extract specific time frames, for example $\boldsymbol{x}_t \in \mathbb{R}^D$, or sub-sequences $\boldsymbol{x}_{1:t} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t]$.

3.2. Normalising flows

Normalising flows are flexible generative models that allow both efficient sampling and efficient inference (likelihood computation). The idea is to subject samples from a simple, fixed distribution $\boldsymbol{Z}$ on $\mathbb{R}^D$ to an invertible and differentiable nonlinear transformation (change of variables) $\boldsymbol{f} : \mathbb{R}^D \to \mathbb{R}^D$ to obtain a new, more complex distribution $\boldsymbol{X}$. The transformation $\boldsymbol{f}$ is parameterised by some $\boldsymbol{\theta}$. If this nonlinear transformation has many degrees of freedom, a wide variety of different distributions can be described.

Like in deep learning in general, expressive transformations $\boldsymbol{f}$ are typically constructed by chaining together numerous simpler transformations $\{\boldsymbol{f}_n\}_{n=1}^N$, each of them parameterised by a $\boldsymbol{\theta}_n$ such that $\boldsymbol{\theta} = \{\boldsymbol{\theta}_n\}_{n=1}^N$. We define the observable random variable (RV) $\boldsymbol{X}$, the latent RV $\boldsymbol{Z}$, and intermediate distributions $\boldsymbol{Z}_N$ as follows:

$$\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{z}; \boldsymbol{0}, \boldsymbol{I}) \tag{1}$$

$$\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{z}) = \boldsymbol{f}_1(\boldsymbol{f}_2(\ldots \boldsymbol{f}_N(\boldsymbol{z}))) \tag{2}$$

$$\boldsymbol{z} = \boldsymbol{z}_N \overset{\boldsymbol{f}_N}{\to} \boldsymbol{z}_{N-1} \overset{\boldsymbol{f}_{N-1}}{\to} \ldots \overset{\boldsymbol{f}_2}{\to} \boldsymbol{z}_1 \overset{\boldsymbol{f}_1}{\to} \boldsymbol{z}_0 = \boldsymbol{x} \tag{3}$$

$$\boldsymbol{z}_n(\boldsymbol{x}) = \boldsymbol{f}_n^{-1} \circ \ldots \circ \boldsymbol{f}_1^{-1}(\boldsymbol{x}). \tag{4}$$

The sequence of (inverse) transformations $\boldsymbol{f}_n^{-1}$ in (4) is known as a *normalising flow*, since it transforms $\boldsymbol{X}$ into a standard normal RV $\boldsymbol{Z}$.

Similar to the generators in GANs, normalising flows are *implicit* generative models as defined in [34], in that they are defined not by a probability density function in the space of the observations $\boldsymbol{X}$ but as a nonlinear transformation $\boldsymbol{f}$ of a latent distribution $\boldsymbol{Z}$. Different from GANs, however, normalising flows permit tractable and efficient inference: Using the change-of-variables formula, we can write the log-likelihood of a sample $\boldsymbol{x}$ as

$$\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \ln p_{\mathcal{N}}(\boldsymbol{z}_N(\boldsymbol{x})) + \sum_{n=1}^{N} \ln \left| \det \frac{\partial \boldsymbol{z}_n(\boldsymbol{x})}{\partial \boldsymbol{z}_{n-1}} \right|, \tag{5}$$

where $\frac{\partial \boldsymbol{z}_n(\boldsymbol{x})}{\partial \boldsymbol{z}_{n-1}}$ is the Jacobian matrix of $\boldsymbol{f}_n^{-1}$ at $\boldsymbol{x}$, which depends on the parameters $\boldsymbol{\theta}$. In the most general case, the determinant in (5) has computational complexity $\mathcal{O}(D^3)$ with many standard algorithms. Several improvements in normalising flows in recent years have concerned the development of invertible, differentiable transformations with tractable (typically triangular) Jacobian matrices, that nonetheless yield highly flexible transformations under iterated composition. In this work, we rely on *Glow* [26], first developed for images, to model motion sequences.

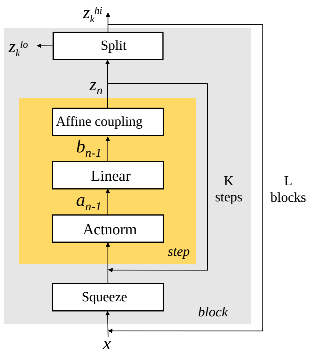

Figure 1. Glow steps $\boldsymbol{f}_n^{-1}$ and hierarchy during inference.

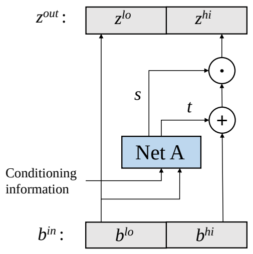

Figure 2. Detail of affine coupling layer in Fig. 1. Conditioning information can be adjacent $\boldsymbol{b}^{\mathrm{lo}}$-values (8), plus control (10), or autoregressive history and control (18).

3.3. Glow

Each component transformation $\boldsymbol{f}_n^{-1}$ in Glow contains three sub-steps: *actnorm*; a convolved *linear transformation*; and a so-called *affine coupling layer*, together shown as a *step* of flow in in Fig. 1. The affine coupling performs an invertible nonlinear transformation of half of the variables based on the values of the other half. Since these remaining variables are passed through unchanged, it is easy to use their values undo the transformation when reversing the computation. The linear layer essentially permutes the variables between couplings, so that all variables (not just one half) are transformed by the full flow. Actnorm is merely intended as a substitute for batchnorm. We will now describe the three sub-steps in the context of time sequences $\boldsymbol{x} \in \mathbb{R}^{T \times D}$, instead of images $\boldsymbol{x} \in \mathbb{R}^{W \times H \times D}$ (with $D = 3$ for RGB) as in the original Glow paper [26], using $\boldsymbol{a}_n$ and $\boldsymbol{b}_n$ to denote intermediate results of the computations.

3.3.1 Actnorm

Actnorm, the first Glow sub-step, is an affine transformation $\boldsymbol{a}_{t,n} = \boldsymbol{s}_n \odot \boldsymbol{z}_{t,n} + \boldsymbol{t}_n$ (where $\odot$ denotes elementwise multiplication), initialised such that the output has zero mean and unit variance on an initial minibatch of data, to mimic batchnorm. $\boldsymbol{s}_n > 0$ and $\boldsymbol{t}_n$ are then treated as trainable parameters (elements of $\boldsymbol{\theta}_n$) during the optimisation.

3.3.2 Linear transformation

This sub-step performs a linear transformation $\boldsymbol{b}_{t,n} = \boldsymbol{W}_n \boldsymbol{a}_{t,n}$ where $\boldsymbol{W} \in \mathbb{R}^{D \times D}$. This is applied to isolated groups of D variables through a convolution with filter-size one. By parameterising and storing $\boldsymbol{W}_n$ as an LU-decomposition $\boldsymbol{W}_n = \boldsymbol{L}_n \boldsymbol{U}_n$ where one of the diagonals (say that of $\boldsymbol{L}_n$) is constrained to contain only ones, the Jacobian log-determinant of the transformation is just the sum of the diagonal elements $\boldsymbol{u}_{n,dd}$, which is computable in linear time. The non-fixed elements of $\boldsymbol{L}_n$ and $\boldsymbol{U}_n$ are the trainable parameters of the sub-step and elements of $\boldsymbol{\theta}_n$.

This linear sub-step can be seen as a differentiable, learnable generalisation of a permutation operation (or of the reversing operation in RealNVP [8]), especially since we initialise it to be an orthogonal transformation. Permutation or mixing is crucial because the subsequent affine coupling layer greatly depends on the order of the variables.

3.3.3 Affine coupling layer

Defining $\boldsymbol{b}_{t,n}$ and $\boldsymbol{z}_{t,n+1}$ as concatenations $\boldsymbol{b}_{t,n} = [\boldsymbol{b}_{t,n}^{\mathrm{lo}}, \boldsymbol{b}_{t,n}^{\mathrm{hi}}]$ and $\boldsymbol{z}_{t,n+1} = [\boldsymbol{z}_{t,n+1}^{\mathrm{lo}}, \boldsymbol{z}_{t,n+1}^{\mathrm{hi}}]$, the affine coupling layer of Glow can be written as

$$\left[\boldsymbol{z}_{t,n+1}^{\mathrm{lo}}, \boldsymbol{z}_{t,n+1}^{\mathrm{hi}}\right] = \left[\boldsymbol{b}_{t,n}^{\mathrm{lo}}, \left(\boldsymbol{b}_{t,n}^{\mathrm{hi}} + \boldsymbol{t}_{t,n}'\right) \odot \boldsymbol{s}_{t,n}'\right], \quad (6)$$

where the scaling $\boldsymbol{s}_n' > \boldsymbol{0}$ and bias $\boldsymbol{t}_n'$ terms in the affine transformation of the variables $\boldsymbol{b}_{t,n}^{\mathrm{hi}}$ are computed via a neural network A_n that only takes $\boldsymbol{b}_{t,n}^{\mathrm{lo}}$ as input:

$$\left[\boldsymbol{s}_{t,n}', \boldsymbol{t}_{t,n}'\right] = A_n\left(\boldsymbol{b}_{t,n}^{\mathrm{lo}}\right). \quad (7)$$

(We use 'A' for "affine".) The computations for this coupling during is inference are visualised in Fig. 2. Since $\boldsymbol{z}_{t,n+1}^{\mathrm{lo}} = \boldsymbol{b}_{t,n}^{\mathrm{lo}}$, the scale and bias terms needed to invert the affine coupling layer are computable unambiguously from $\boldsymbol{z}_{t,n+1}$ also by forward propagation through A_n.

The weights that define A_n are the final elements of the parameter set $\boldsymbol{\theta}_n$. The constraint $\boldsymbol{s}_n' > 0$ is enforced by applying a sigmoid nonlinearity to the corresponding outputs [36, App. D]. The network is initialised with random weights except in the output layer, which is initialised to zero in such a way that the resulting affine transformation is close to an identity transformation. Dependencies between $\boldsymbol{Z}_{t,n}$-values at different times t are introudced by m aking A_n a convolutional neural network (CNN) as in

$$\left[\boldsymbol{s}_{t,n}', \boldsymbol{t}_{t,n}'\right] = A_n\left(\boldsymbol{b}_{t-1:t+1,n}^{\mathrm{lo}}\right). \quad (8)$$

3.3.4 Hierarchical decomposition

Without the affine coupling layer, any chain of flow steps would be equivalent to a single affine transformation. However, because the affine coupling layer depends highly nonlinearly on $\boldsymbol{b}_{t,n+1}^{\mathrm{lo}}$, iterated compositions of the full flow $\boldsymbol{f}_n^{-1}$ above can describe very complex transformations.

To increase the power to model long-range dependencies with limited computation, it is standard to use a hierarchical decomposition of the sequence $\boldsymbol{z}$ [8, 26]. If we let $\boldsymbol{z}_n \in \mathbb{R}^{T_n \times D_n}$ define the size of $\boldsymbol{z}_n$, then for every K steps of flow, half of the $\boldsymbol{z}$-values – $z_{Kl}^{\mathrm{lo}} \in \mathbb{R}^{T_n \times \lfloor D_n/2 \rfloor}$ – are not considered in any further flow steps, but simply passed directly into $\boldsymbol{z}_N$. The other half is squeezed (reshaped) to half the time resolution but without substantially growing the number of flow channels D_{n+1} (to keep the amount of computation manageable), as $z_{Kl+1} \in \mathbb{R}^{T_n/2 \times 2\lceil D_n/2 \rceil}$. This is repeated over L *blocks* (levels in the hierarchy) with K steps of flow in each, as shown in Fig. 1. Putting it all together, the log-likelihood from (5) of a full Glow hierarchy applied to a single sequence $\boldsymbol{x}$ can be written [26, 38]

$$\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \text{const.} - \frac{1}{2}\boldsymbol{z}_N^{\intercal}(\boldsymbol{x})\,\boldsymbol{z}_N(\boldsymbol{x}) + \sum_{n=1}^{N}\sum_{d=1}^{D_n}\sum_{t=1}^{T_n}\left(\ln s_{n\,d} + \ln u_{n\,dd} + \ln s'_{t,\,n,\,d}(\boldsymbol{x})\right), \quad (9)$$

where we have made explicit which terms depend on $\boldsymbol{x}$. It is straightforward to maximise this likelihood in machine-learning frameworks like TensorFlow or Torch.

3.4. Fixed-length models with control

The previous section described how Glow from [26] can be applied to one-dimensional sequences (time series) of vectors. We are particularly interested in describing motion, which typically is represented as a sequence of poses $\boldsymbol{x}_t \in \mathbb{R}^D$ registered at regular intervals (frames). "Poses" can here refer to a whole body, parts of a body, or keypoints on a body or face, in two or three dimensions. The D dimensions commonly encode each pose as joint rotations or Euclidean coordinates of joints or other keypoints (like in the experiments in Sec. 5), although movement in space requires also specifying the position and orientation of the root node(s) in the hierarchy. Applying one-dimensional Glow out of the box to such pose sequences yields an *unconditional* model of motion – that is, a description of random plausible motions resembling those in one's training database, but without any ability to exert control.

Unfortunately, unconditional models are seldom enough. In the vast majority of data-generation (synthesis) applications, output does not only need to be natural, but it is vital to also be able to make it satisfy certain constraints defined by the application. In motion applications, one might, e.g., want locomotion to follow a certain path, body language to express a certain emotion, or face and body motion to match a spoken message. Mathematically, control can be realised by learning the distribution of $\boldsymbol{X}$ conditioned on a *control signal* $\boldsymbol{C}$ (another random variable). We assume that, for each training-data frame $\boldsymbol{x}_t$, the matching control-signal values $\boldsymbol{c}_t \in \mathbb{R}^C$ are known. "Global" control parameters that are constant across an entire output sequence can easily be represented in this framework by including elements in $\boldsymbol{c}_t$ that are kept fixed for each sequence.

In theory, a joint unconditional distribution implicitly describes all conditional and unconditional distributions of the component variables through the (completely general) factorisation $p(\boldsymbol{x},\,\boldsymbol{y}) = p(\boldsymbol{x}\,|\,\boldsymbol{y})\,p(\boldsymbol{y})$. Unfortunately, there are currently no established, tractable methods for sampling from conditional distributions $p(\boldsymbol{x}\,|\,\boldsymbol{c})$ defined by an unconditional normalising flow $p(\boldsymbol{x},\,\boldsymbol{c})$. To ensure precise control over output motion, we propose to learn the conditional data distribution $p(\boldsymbol{x}\,|\,\boldsymbol{c})$ from the start. For this, we look to the proposal in the recent papers [38, 24] on Glow for audio waveforms controllable by mel-spectrogram features. The key modification is to let the input to A_n in the affine coupling layer at t depend not only on (a contextual window) around $\boldsymbol{b}^{\text{lo}}_{t,\,n}$ but also on the local control input $\boldsymbol{c}_{t,\,n}$[1], as in

$$\left[\boldsymbol{s}'_{t,\,n},\,\boldsymbol{t}'_{t,\,n}\right] = A_n\left(\boldsymbol{b}^{\text{lo}}_{t-1:t+1,\,n},\,\boldsymbol{c}_{t-1:t+1,\,n}\right). \quad (10)$$

Sub-sequences of $\boldsymbol{c}$ thus act as conditioning input to A_n in Fig. 2. The log-likelihood terms in (9) that depend on $\boldsymbol{x}$ now also depend on $\boldsymbol{c}$, but optimising this augmented likelihood presents no conceptual difficulty.

4. MoGlow: Glow for motion

The motion models described in Sec. 3.4 are limited to describing fixed-length motion sequences generated in one single operation. In this section we describe our new variant of Glow that is designed to model vector-valued sequences, including motion, in a stepwise and incremental manner. Our model simplifies the modelling of temporal dependencies in WaveGlow [38] and WaveFlowNet [24] by using recurrent neural networks (RNNs) instead of dilated convolutions. This is consistent with recent developments in generative modelling of waveform audio, which is seeing a shift away from dilated convolutions as introduced in WaveNet [50] towards more computationally attractive recurrent approaches such as WaveRNN [22] and its relatives [49, 31]. Unlike the similar VideoFlow [29] model we also consider external control of the generated sequences.

4.1. Autoregressive models

The starting point for building autoregressive models of time sequences is the (always valid) decomposition

$$p(\boldsymbol{x}) = p(\boldsymbol{x}_{1:\tau})\prod_{t=\tau+1}^{T} p(\boldsymbol{x}_t\,|\,\boldsymbol{x}_{1:t-1}) \quad (11)$$

If one assumes that the distribution of $\boldsymbol{x}_t$ only depends on the τ previous values, then one obtains

$$p_{\text{Markov}}(\boldsymbol{x}) = p(\boldsymbol{x}_{1:\tau})\prod_{t=\tau+1}^{T} p(\boldsymbol{x}_t\,|\,\boldsymbol{x}_{t-\tau:t-1}), \quad (12)$$

[1]We write $\boldsymbol{c}_{t,\,n}$ since the time-resolution of $\boldsymbol{c}$ will be affected by the squeezing in the hierarchy from Sec. 3.3.4. As $\boldsymbol{c}$ is never split, the dimensionality of $\boldsymbol{c}_t$ doubles with each block. However, this only affects the computational complexity of A_n in (10), not that of the coupling in (6), and is not a problem if the control signal dimensionality C is modest.

a process with finite-length memory known as a *Markov chain*. We call $p\left(\boldsymbol{x}_t \,|\, \boldsymbol{x}_{t-\tau:t-1}\right)$ the *next-step distribution*.

To allow longer memory, one can introduce a hidden (unobservable) state $\boldsymbol{h}_t \in \mathbb{R}^H$ which evolves according to a function $\boldsymbol{g}$ at each timestep and influences the observable distribution according to the model

$$p_{\text{state}}\left(\boldsymbol{x}\right) = p\left(\boldsymbol{x}_{1:\tau}\right) \prod_{t=\tau+1}^{T} p\left(\boldsymbol{x}_t \,|\, \boldsymbol{x}_{t-\tau:t-1},\, \boldsymbol{h}_t\right) \quad (13)$$

$$\boldsymbol{h}_{t+1} = \boldsymbol{g}\left(\boldsymbol{x}_{t-\tau:t-1},\, \boldsymbol{h}_t\right) \quad (14)$$

with $\boldsymbol{h}_\tau$ here initialised to $\mathbf{0}$.[2] We call this a (hidden) *state-space model*. This model has complete memory τ steps back in time, but can also model dependencies further back in time (theoretically all the way back to $\boldsymbol{x}_1$) thanks to the hidden state variable. For deterministic $\boldsymbol{g}$ a straightforward choice to implement (14) is to use a recurrent neural network, for instance an LSTM [16].

For many processes, including motion, it is reasonable to assume that the laws that govern the process are the same at any given point in time, meaning that the next step-distribution is independent of t. That is,

$$p\left(\boldsymbol{x} \,|\, \boldsymbol{x}_{t-\tau:t-1},\, \boldsymbol{h}_t\right) = p\left(\boldsymbol{x} \,|\, \boldsymbol{x}_{t'-\tau:t'-1},\, \boldsymbol{h}_{t'}\right) \quad (15)$$

whenever $\boldsymbol{x}_{t-\tau:t-1} = \boldsymbol{x}_{t'-\tau:t'-1}$ and $\boldsymbol{h}_t = \boldsymbol{h}_{t'}$. This is known as *stationarity* and is an exceedingly common assumption in practical sequence modelling. Since each time step in the training data is an example of the same time-independent conditional distributions, stationary models are straightforward and efficient to train.

4.2. Conditioning on history and control

The basic idea of our autoregressive Glow model is to learn the next-step distribution $p\left(\boldsymbol{x}_t \,|\, \boldsymbol{x}_{t-\tau:t-1},\, \boldsymbol{h}_t\right)$ using Glow. This models a C-dimensional distribution that generates individual frames of motion iteratively in time, with a separate latent $\boldsymbol{z}_t$ for each time. A window $\boldsymbol{x}_{t-\tau:t-1}$ of motion (poses from preceding frames) is used to condition the affine coupling, analogous with how $\boldsymbol{X}$ was conditioned on $\boldsymbol{C}$ in Sec. 3.4. This is equivalent to conditioning on $\boldsymbol{z}_{t-\tau:t-1}$ since the mapping $\boldsymbol{x}_t = \boldsymbol{f}\left(\boldsymbol{z}_t\right)$ is invertible.

Since there is no translation invariance along the C-dimension and only one frame is generated at a time in this autoregressive approach, we achieved better results by not making the network in the affine coupling convolutional (although the net is applied to in a convolutional manner across time, owing to the stationarity). However, for the τ-values we considered in the experiments (on the order of five or ten frames), we found a (hidden) state-space models with LSTMs substantial superior to Markovian models where A_n was a feedforward neural net. Specifically, using LSTMs improved the likelihoods and the subjective naturalness of the motion, and also appeared more stable to train. We only consider hidden state-space (i.e., recurrent) models for the autoregressive models in the experiments of this article.

[2]We will ignore how to model the initial distribution $p\left(\boldsymbol{x}_{1:\tau}\right)$ from (13) in this article and will exclude its contribution to any probability computations; when generating output we will initialise models with $\boldsymbol{h}_{\tau+1} = \mathbf{0}$ and ground-truth motion segments $\boldsymbol{x}_{1:\tau}$ from the database.

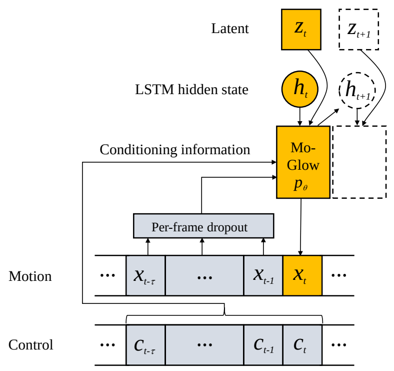

Figure 3. Schematic illustration of autoregressive motion generation with MoGlow. Dropout is only applied at training time.

Like in Sec. 3.4, we also wish to use an aligned sequence of control inputs $\boldsymbol{c}$ to influence the motion, e.g., to make locomotion follow a given path through space. Our goal is for this control to be possible to apply in real time without latency. For this reason, the distribution of $\boldsymbol{x}_t$ can only depend on current and past control inputs $\boldsymbol{c}_{1:t}$, but not on $\boldsymbol{c}$-values for times greater than t. We thus arrive at a model

$$p_{\boldsymbol{\theta}}\left(\boldsymbol{x} \,|\, \boldsymbol{c}\right) = p\left(\boldsymbol{x}_{1:\tau} \,|\, \boldsymbol{c}_{1:\tau}\right) \cdot \prod_{t=\tau+1}^{T} p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_t \,|\, \boldsymbol{x}_{t-\tau:t-1},\, \boldsymbol{c}_{t-\tau:t},\, \boldsymbol{h}_t\right) \quad (16)$$

$$\boldsymbol{h}_{t+1} = \boldsymbol{g}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t-\tau:t-1},\, \boldsymbol{c}_{t-\tau:t},\, \boldsymbol{h}_t\right) \quad (17)$$

$$\left[\boldsymbol{s}'_{t,\,n},\, \boldsymbol{t}'_{t,\,n}\right] = A_n\left(\boldsymbol{b}^{\text{lo}}_{t,\,n},\, \boldsymbol{x}_{t-\tau:t-1},\, \boldsymbol{c}_{t-\tau:t,\,n},\, \boldsymbol{h}_t\right), \quad (18)$$

where the next-step distribution in (16) is implemented by means of a Glow-style normalising flow with LSTMs in the affine coupling layer that take $\boldsymbol{x}_{t-\tau:t-1}$ and $\boldsymbol{c}_{t-\tau:t}$ as conditioning input in (18). Eq. (17) is realised through the evolution of the LSTM state. A schematic illustration of sample generation with this model is presented in Fig. 3. We call models with this structure *MoGlow* for *motion Glow*.

Our implementation of MoGlow contains no squeeze operations, although there might be reasons to consider squeezing, especially at high frame rates (cf. the discussion in Sec. 5.3). We did not notice any benefits of a hierarchical decomposition within individual frames $\boldsymbol{x}_t$ in preliminary experiments, so we set $L = 1$ for these models, producing a simple yet powerful architecture.

5. Experiments

This section describes an application – and subsequent evaluation – of the methods from Secs. 3 and 4 on a motion-capture dataset of human locomotion. We stress that, unlike, e.g., [17], there is nothing in our approach that requires the motion to be (quasi-)periodic like human gait.

5.1. Data

We evaluated our methods on the locomotion trials from the CMU and HDM05 [35] motion-capture datasets pooled with the Edinburgh locomotion database [15]. The motion was retargeted by [19, 18] to the same-size skeleton and represents an actor walking (backwards, forwards, and side-stepping) and running in different directions on a flat surface. We held out a subset of the data with a roughly equal amount of motions in each category for evaluation, and used the rest for training. We sub-sampled the data from 120 to 20 frames per second and sliced it into fixed-length 4-second windows, resulting in 13,710 training sequences. The downsampling reduces both computation and the exposure bias in the MoGlow model, as discussed in Sec. 5.3.

Each frame in the database consists of 3D Euclidean coordinates for 21 joint positions (63 degrees of freedom) on a skeleton expressed in a root coordinate system projected onto the floor plus the forwards, sideways, and angular (around the up-axis) displacement of the root. We used the 63-dimensional skeleton pose as the output $\boldsymbol{x}_t$, treating the three displacement variables as the control $\boldsymbol{c}_t$. In other words, the control signal defines a 2D track $\boldsymbol{c}$ through space along which motion occurs with a given speed and rotation; the models are supposed to complete the motion with an appropriate sequence $\boldsymbol{x}$ of body poses. The data was augmented by mirroring. In addition, it was found that very few sequences in the data contained backwards and side-stepping motion. To increase the amount of data expressing such motion, the dataset was also mirrored in time. A preliminary comparison confirmed that this time-reversal substantially improved the naturalness of synthesised motion.

5.2. Model setup and training

We trained[3] a fixed-length model as in Sec. 3.4 with $L = 4$ blocks of $K = 32$ steps of flow each, not splitting (but squeezing) $\boldsymbol{c}_t$-vectors between each block. The neural network in the affine coupling layers used two hidden convolutional layers with 512 nodes and ReLU nonlinearities. We also trained an autoregressive MoGlow model as in Sec. 4.2, using a $\tau = 10$ frame time-window (0.5 seconds) with $L = 1$ block of $K = 16$ steps and 2 LSTM layers of 512 nodes each.

Each dimension in the data and the control signal was standardised to zero mean and unit variance before training. Optimisation maximised the log-likelihood of the training-data sequences using Adam [25]. The fixed-length model was trained for 20k steps and the autoregressive model for 80k steps. Despite the larger number of steps, the latter model required less wall-clock time due to its smaller size. We note that training consistently "just worked" without the need for tuning optimiser hyperparameters, in contrast to theoretical [33] and practical [32] issues identified with many GAN training paradigms.

5.3. Dropout on autoregressive inputs

A practical issue with optimising autoregressive models of slowly-changing sequences is that the optimisation might not successfully integrate all the pertinent information from the input, for instance getting stuck on predicting $\boldsymbol{x}_t = \boldsymbol{x}_{t-1}$, while ignoring other inputs such as the control signal. This is a common stumbling block in generative models of speech feature sequences, as evidenced in [48, 52, 44]. Established methods to counter this failure mode of the optimisation include applying dropout to frames of autoregressive history inputs, as in [4, 52], or downsampling the data sequences as in [44]. Both of these have the net effect of making the (on average) most-recent autoregressive input-frame available to the network be further removed from the current output time, meaning that the information value of the autoregressive feedback is decreased, and the information in the current control input becomes relatively more valuable. The squeeze operations in [8, 26, 38, 24] can be seen as another way to reduce the information shared between adjacent outputs, although only [38, 24] apply these squeezes in an autoregressive model.

We found that introducing dropout substantially improved the consistency between the generated motion and the control signal. Without dropout, generated motion often walked or ran even when the control signal indicated that no movement through space was taking place. Dropout was also found to remedy issues with exposure bias [40]: While early autoregressive MoGlow models tended to revert towards a static pose when sampling sequences several seconds in duration, this issue virtually disappeared after applying dropout to the autoregressive history inputs. For our experiments we set the per-frame dropout rate to 0.95.

5.4. Subjective evaluation

In order to assess the naturalness of the generated locomotion sequences we conducted a subjective evaluation, were evaluators were asked to rate the naturalness of a number of short (4-second) animation clips generated by our systems, in which motion was visualised using a stick-figure (a so-called *skeleton*) seen from a fixed camera angle; see Fig. 4. Natural and synthetic motion examples can be seen in our presentation video at youtu.be/lYhJnDBWyeo.

In the experiment we compared sequences generated

[3]All our code was based on github.com/chaiyujin/glow-pytorch.

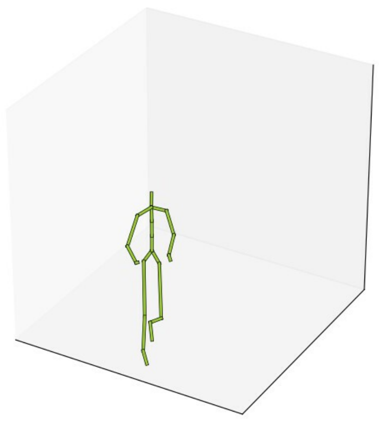

Figure 4. Still image from video used in the subjective evaluation.

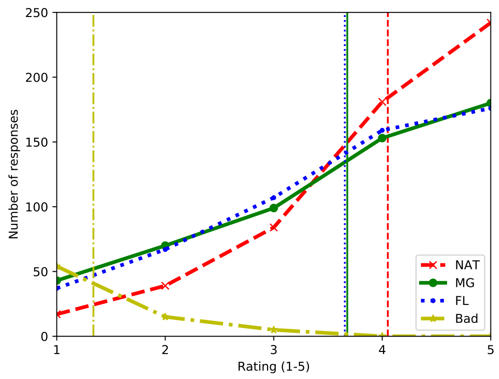

Figure 5. Histogram of responses on the five-point scale for each condition. Vertical lines signify mean ratings for the conditions.

from the fixed-length model, referred to as **FL** below, and the autoregressive MoGlow model, referred to as **MG** below, against ground-truth sequences from the mocap system, which we label **NAT** for natural. We used the *Figure Eight* crowd worker platform, with the highest-quality contributor setting (allowing only the most experienced, highest-accuracy contributors). The experiment contained 94 animation clips: 30 animations clips for each of the three conditions, plus 4 examples of *bad* animation taken from early iterations in the training process. These were added as a control measure to be able to filter out unreliable raters.

Raters were asked to grade the perceived naturalness of each animation on a scale from 1 to 5, where 1 is *completely unnatural* (motion could not possibly be produced by a real person) and 5 is *completely natural* (looks like the motion of a real person). The order of the animation clips was randomised, and no information was given to the raters about which system had generated a given video, nor how many systems that were being evaluated in the test. Prior to the start of the rating, subjects were trained by viewing example motion videos from the different conditions evaluated, as well as some of the bad examples mentioned above.

Each of the 94 animations was judged by 20 independent raters, yielding a total of 1880 ratings. Any raters that had given the *bad examples* a rating of 3 or higher were discarded from the experiment, causing 152 observations to be removed and leaving 1728.

5.5. Results and discussion

The mean scores for each condition were 4.05 for **NAT**, 3.68 for **MG**, and 3.66 for **FL**. (For comparison, the *bad example* control condition received a mean score of 1.34, however these data points were not used in the rest of the analysis.) Fig. 5 reveals the distribution of responses for the different conditions. It can be noted that the most common rating for all three conditions was 5.

A one-way ANOVA revealed a main effect of **NAT**, **MG**, and **FL** ($F = 19.4$, $p < 10^{-8}$), and a post-hoc Tukey multiple comparison test identified a significant difference between **NAT** and **MG**, and between **NAT** and **FL**, but *not* between the two synthetic systems **FL** and **MG** (0.05 FWER).

While the animations produced by the two systems in the experiment were perceived somewhat less natural than the ground-truth motion, it is interesting to note that the quality of the output from the proposed autoregressive system is rated on par with that from the noncausal, fixed-length implementation. The main implication of this is that the advantages of the autoregressive method, i.e., the ability to generate animation continuously on the fly – enabling applications such as real-time control of computer game-characters, robots, or virtual actors – can be gained without sacrificing motion quality.

6. Conclusion and future work

We have described the first models of motion-data sequences based on normalising flows. Flows attractive because they i) are probabilistic (unlike many established motion models), ii) utilise powerful, implicitly defined distributions (like GANs but unlike classical autoregressive models), yet iii) can be trained to directly maximise data likelihood (unlike GANs and VAEs). Both unconditional and conditional (i.e., controllable) models have been described.

Our flagship model, dubbed MoGlow, uses autoregression and a hidden state (recurrence) to allow sequential output generation that can be controlled without latency. To our knowledge, no other Glow-based sequence models combine these desirable traits, and no other such model has incorporated hidden states or autoregression dropout for better control. In a subjective evaluation of human locomotion generation, the rated quality of randomly-sampled motions was numerically close to that of natural motion, with no significant difference between fixed-length models closer to the original Glow [26] and our faster, simpler, sequential, and latency-free MoGlow model.

Considering the quality of the generated motion and the generally-applicable nature of the approach, we believe that models based on normalising flows will prove valuable for a wide variety of tasks incorporating motion data. Future work includes expanded experiments that apply the methods to several motion types and evaluate them both objectively and subjectively. Since models based on normalising flows allow exact and tractable inference, another interesting future application is to use the probabilities inferred by these models in decision tasks such as classification.

References

[1] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *Proc. ICLR*, 2019. 2

[2] J. Bütepage, M. J. Black, D. Kragic, and H. Kjellström. Deep representation learning for human motion prediction and classification. In *Proc. CVPR*, pages 1591–1599, 2017. 3

[3] J. Chai and J. K. Hodgins. Performance animation from low-dimensional control signals. *ACM T. Graphic.*, 24(3):686–696, 2005. 3

[4] X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. Variational lossy autoencoder. In *Proc. ICLR 2017*, 2017. 7

[5] C. Cremer, X. Li, and D. Duvenaud. Inference suboptimality in variational autoencoders. In *Proc. ICLR Workshop*, 2018. 2

[6] G. Deco and W. Brauer. Higher order statistical decorrelation without information loss. In *Proc. NIPS*, pages 247–254, 1995. 1, 2

[7] L. Dinh, D. Krueger, and Y. Bengio. NICE: Non-linear independent components estimation. In *Proc. ICLR Workshop*, 2015. 1, 2

[8] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP. In *Proc. ICLR*, 2017. 1, 2, 4, 7

[9] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent network models for human dynamics. In *Proc. CVPR*, pages 4346–4354, 2015. 3

[10] I. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016. 2

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proc. NIPS*, pages 2672–2680, 2014. 2

[12] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013. 2

[13] D. Greenwood, S. Laycock, and I. Matthews. Predicting head pose from speech with a conditional variational autoencoder. In *Proc. Interspeech*, pages 3991–3995, 2017. 3

[14] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović. Style-based inverse kinematics. *ACM T. Graphic.*, 23(3):522–531, 2004. 3

[15] I. Habibie, D. Holden, J. Schwarz, J. Yearsley, and T. Komura. A recurrent variational autoencoder for human motion synthesis. In *Proc. BMVC*, 2017. 3, 7

[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997. 6

[17] D. Holden, T. Komura, and J. Saito. Phase-functioned neural networks for character control. *ACM T. Graphic.*, 36(4):42:1–42:13, 2017. 2, 3, 7

[18] D. Holden, J. Saito, and T. Komura. A deep learning framework for character motion synthesis and editing. *ACM T. Graphic.*, 35(4):138:1–138:11, 2016. 3, 7

[19] D. Holden, J. Saito, T. Komura, and T. Joyce. Learning motion manifolds with convolutional autoencoders. In *Proc. SIGGRAPH Asia Technical Briefs*, pages 18:1–18:4, 2015. 3, 7

[20] F. Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*, 2015. 2

[21] F. Huszár. Is maximum likelihood useful for representation learning? http://www.inference.vc/maximum-likelihood-for-representation-learning-2/, 2017. 2

[22] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu. Efficient neural audio synthesis. In *Proc. ICML*, pages 2410–2419, 2018. 2, 5

[23] T. Karras, T. Aila, S. Laine, A. Herva, and J. Lehtinen. Audio-driven facial animation by joint end-to-end learning of pose and emotion. *ACM T. Graphic.*, 36(4):94, 2017. 3

[24] S. Kim, S.-g. Lee, J. Song, and S. Yoon. FloWaveNet: A generative flow for raw audio. *arXiv preprint arXiv:1811.02155*, 2018. 2, 5, 7

[25] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015. 7

[26] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Proc. NeurIPS*, pages 10236–10245, 2018. 1, 2, 3, 4, 5, 7, 8

[27] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *Proc. ICLR*, 2014. 2

[28] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. *ACM T. Graphic.*, 21(3):473–482, 2002. 3

[29] M. Kumar, M. Babaeizadeh, D. Erhan, C. Finn, S. Levine, L. Dinh, and D. Kingma. VideoFlow: A flow-based generative model for video. *arXiv preprint arXiv:1903.01434*, 2019. 2, 5

[30] S. Levine, J. M. Wang, A. Haraux, Z. Popović, and V. Koltun. Continuous character control with low-dimensional embeddings. *ACM T. Graphic.*, 31(4):28, 2012. 3

[31] J. Lorenzo-Trueba, T. Drugman, J. Latorre, T. Merritt, B. Putrycz, and R. Barra-Chicote. Robust universal neural vocoding. *arXiv preprint arXiv:1811.06292*, 2018. 5

[32] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are GANs created equal? A large-scale study. In *Proc. NeurIPS*, pages 698–707, 2018. 2, 7

[33] L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for GANs do actually converge? In *Proc. ICML*, pages 3481–3490, 2018. 2, 7

[34] S. Mohamed and B. Lakshminarayanan. Learning in implicit generative models. In *Proc. ICLR Workshop*, 2017. 2, 3

[35] M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, and A. Weber. Documentation mocap database HDM05. Technical Report CG-2007-2, Universität Bonn, 2007. 7

[36] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan. Do deep generative models know what they don't know? In *Proc. ICLR*, 2018. 4

[37] D. Pavllo, D. Grangier, and M. Auli. QuaterNet: A quaternion-based recurrent model for human motion. In *Proc. BMVC*, 2018. 3

[38] R. Prenger, R. Valle, and B. Catanzaro. WaveGlow: A flow-based generative network for speech synthesis. *arXiv preprint arXiv:1811.00002*, 2018. 2, 5, 7

[39] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286, 1989. 2

[40] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. In *Proc. ICLR*, 2016. 7

[41] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proc. ICML*, pages 1278–1286, 2014. 2

[42] P. Rubenstein. Variational autoencoders are not autoencoders. http://paulrubenstein.co.uk/variational-autoencoders-are-not-autoencoders/, 2019. 2

[43] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications. In *Proc. ICLR*, 2017. 2

[44] H. Tachibana, K. Uenoyama, and S. Aihara. Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention. In *Proc. ICASSP*, pages 4784–4788, 2018. 7

[45] S. Taylor, T. Kim, Y. Yue, M. Mahler, J. Krahe, A. G. Rodriguez, J. Hodgins, and I. Matthews. A deep learning approach for generalized speech animation. *ACM T. Graphic.*, 36(4):93, 2017. 3

[46] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. *Proc. ICLR*, 2016. 2

[47] B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle. Neural autoregressive distribution estimation. *J. Mach. Learn. Res.*, 17(1):7184–7220, 2016. 2

[48] B. Uria, I. Murray, S. Renals, C. Valentini-Botinhao, and J. Bridle. Modelling acoustic feature dependencies with artificial neural networks: Trajectory-RNADE. In *Proc. ICASSP*, pages 4465–4469, 2015. 2, 7

[49] J.-M. Valin and J. Skoglund. LPCNet: Improving neural speech synthesis through linear prediction. In *Proc. ICASSP*, 2019. 5

[50] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016. 2, 5

[51] K. Vougioukas, S. Petridis, and M. Pantic. End-to-end speech-driven facial animation with temporal GANs. *arXiv preprint arXiv:1805.09313*, 2018. 3

[52] X. Wang, S. Takaki, and J. Yamagishi. Autoregressive neural f0 model for statistical parametric speech synthesis. *IEEE/ACM T. Audio Speech*, 26(8):1406–1419, 2018. 7

[53] H. Zen and A. Senior. Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis. In *Proc. ICASSP*, pages 3844–3848, 2014. 2

A. Additional analysis

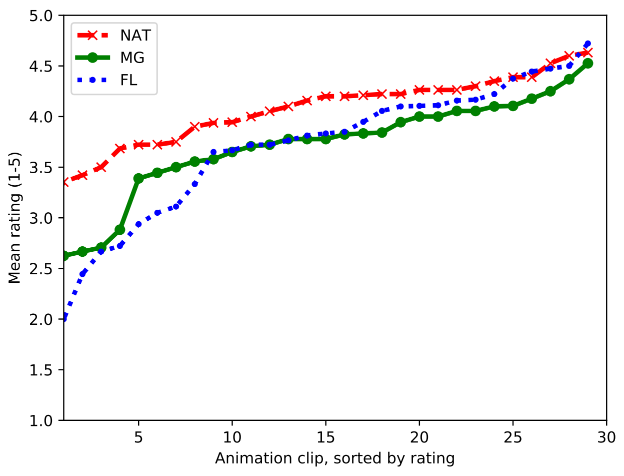

Figure 6. Average rater score for each animation clip, sorted by rating (worst to best) for each system.

Since MoGlow motion-output generation is stochastic – it involves random sampling from a distribution – the generated motion will not be the same every time even for a given, fixed control signal. In particular, the quality of the motion may fluctuate from sample to sample, in addition to the effect that different control signals. In applications with a human in the loop, such as animation, this functionality can be leveraged to the animator's advantage by generating several motion candidates and picking the one that appeals the most to the observer.

To visualise the span of different ratings and their dependence on the motion-control trajectory (the control input C), the plots in Fig. 6 show the distribution of per-item average scores for the conditions NAT, MG, and FL. It is seen that the best synthetic sequences are virtually on par with NAT, but that a few (about 15 or 30%) less strong examples bring down the overall the average score of MG and FL. We also observe that many synthetic motion sequences score better on average than several of the natural (NAT) sequences, indicating a notable overlap in mean rating between motion-captured and synthesised motion examples.