A Convolutional Sequence to Sequence Model for Multimodal Dynamics Prediction in Ski Jumps

Dan Zecha, Christian Eggert, Moritz Einfalt, Stephan Brehm, Rainer Lienhart
University of Augsburg
Augsburg, Germany
[dan.zecha,christian.eggert,moritz.einfalt,stephan.brehm,rainer.lienhart]@informatik.uni-augsburg.de

ABSTRACT

A convolutional sequence to sequence model for predicting the jump forces of ski jumpers directly from pose estimates is presented. We collect the footage of multiple, unregistered cameras together with the output of force measurement plates and present a spatiotemporal calibration procedure for all modalities which is merely based on the athlete's pose estimates. The synchronized data is used to train a fully convolutional sequence to sequence network for predicting jump forces directly from the human pose. We demonstrate that the best performing networks produce a mean squared error of 0.062 on normalized force time series while being able to identify the moment of maximal force occurrence in the original video at 55% recall within ± 2 frames around the ground truth.

CCS CONCEPTS

• **Information systems** → **Temporal data**; *Multimedia and multimodal retrieval*; • **Computing methodologies** → **Camera calibration**; **Machine learning**; **Neural networks**;

KEYWORDS

Deep Learning; Convolutional Sequence Modeling; Multimodal Model; Dynamics Prediction; Jump Force Prediction; Temporal Neural Networks

ACM Reference Format:
Dan Zecha, Christian Eggert, Moritz Einfalt, Stephan Brehm, Rainer Lienhart. 2018. A Convolutional Sequence to Sequence Model for Multimodal Dynamics Prediction in Ski Jumps. In *1st International Workshop on Multimedia Content Analysis in Sports (MMSports'18), October 26, 2018, Seoul, Republic of Korea.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3265845.3265855

1 INTRODUCTION

The motion of a ski jumper can be divided into four parts. During the in-run, the athlete accelerates while skiing down a jumping ramp. Reaching the take-off table at the end of the ramp, the athlete launches himself through a jumping motion into the flight phase (Figure 1). During the flight phase, the jumper targets to assume an optimal posture to increase flight time and thereby extend jump distance. The jump is completed with the landing phase, where the final distance is weighted with external factors to yield a final score for the jump. The final distance a ski jumper travels during a jump can be actively influenced during each phase of the jump. An optimal streamlined in-run guarantees enough initial velocity, while a steady and precisely executed jump motion on the take-off table and a stable and optimal flying posture increase the final distance.

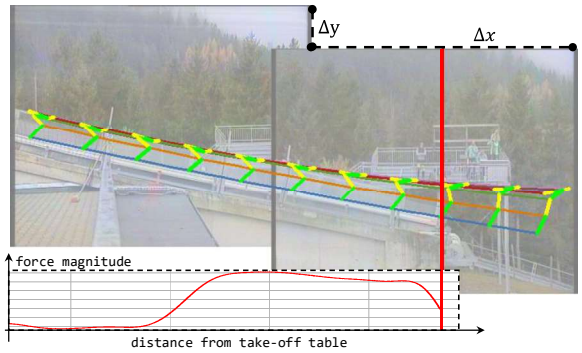

Figure 1: Based on the predicted poses of an athlete (green/yellow), joint trajectories and camera offset Δx and Δy for two camera views can be inferred with the method described in Section 3.2. The red line indicates the edge of the take-off table, which serves as the anchor for force-image synchronization. The synchronized force signal for the jump is depicted as a red graph.

During training sessions, coaches put much effort into optimizing the ideal timing for executing the jump motion on the take-off table. Therefore, ski jump hills are equipped with an array of cameras parallel to the flight path of the athlete. Additionally, force measurement plates are installed on a jumping ramp to precisely measure the jump force during the last meters of the in-run. After a run, coaches review measured force series as well as identify points of interest like the temporal occurrence of the strongest force while precisely annotating and evaluating the athlete's posture. This process is usually done manually and thereby time-consuming. The maintenance of arrays of force measurement plates is also expensive, and the correct function cannot be guaranteed at all times, as the hardware is exposed to extreme weather very often.

In this paper, we strive to automate the whole jump evaluation process using recent advances in computer vision and deep learning. Our final ambition is to develop a system that needs only the

footage of a jump to be able to predict the jump forces of an athlete. As the system will eventually be trained from corresponding pose and force, two problems need to be solved. First, different camera views need to be registered in a mutual coordinate system, followed by temporal synchronization between camera views and force measurements. Second, a model for generating sequences of force predictions from sequences of pose estimates has to be defined.

We propose a solution for both steps as follows. Given some different cameras and the measurements of multiple force measurement plates, we first describe a solution for temporal and spatial registration of all modalities solely based on continuous pose estimates of an athlete. The same estimates are used as input to a purely convolutional sequence to sequence neural network for predicting the jump force.

Our contributions are as follows:

- We are among the first to predict human jump forces directly from pose estimates using a multimodal learning approach, e.g., multiple cameras and force measurement plates.
- The proposed sequence to sequence network structures are strictly convolutional. Similar structures have recently been used for discrete-valued data problems like an automated translation. We demonstrate that with proper network architecture, a convolutional sequence to sequence model can be used for simple regression tasks on strictly continuous input data.
- Our camera registration and force synchronization methodologies are based merely on the motion and flight trajectory of an athlete present in all camera images.

2 RELATED WORK

Camera-calibration. The topic of non-classical camera camera-calibration, i.e., not based on homography estimation between overlapping cameras, was addressed from different directions in the past. [16, 26] propose search algorithms for temporal alignment using human kinematics and point trajectories in dual camera setups. Padua et al. [22] extend the two camera scenario to multi-camera systems. They define a search space of calibration and synchronization parameters called a timeline and proposed an optimization algorithm for approaching the optimal settings. A common feature in all these algorithms is that cameras need to film the same scene to optimize spatiotemporal parameters.

Object dynamics/kinematics. Predicting dynamics of objects in still images is commonly based on neural networks with two input arms, one for a (masked) image and a second for force inputs. Forces are modeled using a game engine [19] or a physics engine [20, 28], where the whole image scene can be simulated. Alternatively, scenarios like billiard games [3, 7] are examined, where a force can be applied to the scene in a controlled environment. Recurrent networks have mainly been used for the prediction of human dynamics from just image/video. For example, deep recurrent neural networks(RNNs) [18] and Encoder-Recurrent-Decoder networks [8] are used to predict the motion of a person continuously. While they often state that the dynamics of a person's movement are estimated, the actual prediction only entails joint kinematics.

Sequence to sequence learning. Sequence to sequence networks have been extensively researched in the last years for machine translation [15], audio synthesis [25] or language modeling [6]. While network structures build on Recurrent Networks [2], LSTMs [12] or Gated Recurrent Units [5] are numerous, convolutional sequence to sequence modeling as an alternative has gained traction recently in machine translation [9] as well as human kinematics prediction [17]. Compared to recurrent models, the width of the history size can be directly defined and controlled through the receptive field of the network. Additionally, computations over all layers can be fully parallelized (in contrary to recurrent models).

Our work. While our work is strongly influenced by generic state-of-the-art networks [1] for time series prediction with convolutional networks, classical applications of these models often work on discrete-valued input and output data. In contrast to that, we attempt to use continuous input data to predict purely continuous output data. The prediction of real-world force measurements solely form camera footage in a entirely uncalibrated scenario - especially within our scope of application, is a novel problem.

3 MULTIMODAL REGISTRATION AND SYNCHRONIZATION

In the introduction, we stated that the process of predicting forces from human poses involves the registration and synchronization of different modalities to train a neural model that performs the actual mapping. The necessity of having to calibrate the modalities is a concern in all multimodal systems. Particularly in ski jumping, the measurement setup has to cover a wide area (the jumping hill) where it is often not feasible to densely equip all corners with sensors and rely on well-studied methods - like extrinsic camera calibration from overlapping camera views - to calibrate the system. Additionally, the measurement equipment is exposed to a variety of weather situations, making re-calibration necessary on a regular basis (even between runs). In the following, we first describe the measurement setup on a fully equipped ski jumping hill. Following this description, we propose a simple yet effective registration and synchronization strategy for all modalities in the system.

3.1 Measurement Setup

A ski jump hill is equipped with N cameras $\mathbf{C} = (C_1, ..., C_N)$ along the take-off table and the flight trajectory of the athlete. All cameras shoot at 50 Hz and are temporally synchronized by the athlete triggering a light barrier when he starts his jump at the top of the in-run. Cameras are regularly spaced in parallel to the fight path and usually do not overlap, except for two cameras with minimal overlap pointing at the take-off table. While all image planes are assumed to be parallel to the flight trajectory of the athlete, we can not guarantee the same distance between each image plane and the jumper due to terrain limitations.

Force measurements are taken at a frequency of 2 kHz from a series of force measurement plates installed at the last 17 meters of the in-run, up to the edge of the take-off table. An external synchronization system for cameras and force plates does not exist. The plates are triggered and synchronized among one another by several light barriers along the in-run. The measurement system compiles the output force signal by accumulating the measurements

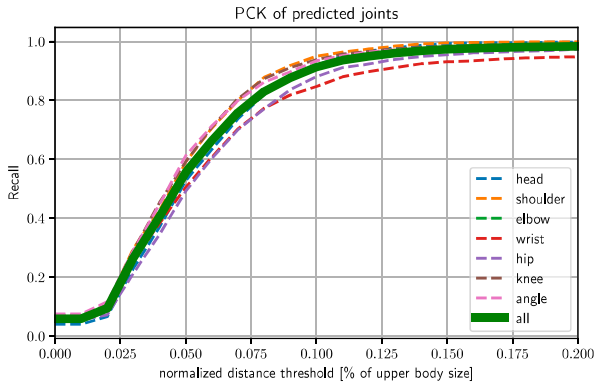

Figure 2: Percentage of correct keypoints [24] for the fine-tuned detector [27] used for pose estimation in this work. We achieve very competitive results (>99% recall @ PCK0.2), which allows for on a par pose estimates as a good starting point for camera calibration and force prediction.

from all plates. Due to the light barrier triggers, it is known at what time and in which distance from the edge of the take-off table a particular force measurement was taken. As a result, the position of the edge together with the location of the jumper (more precisely: his ankle) can be used as a temporal synchronization anchor in both force signal and continuous pose estimates, respectively. Using the position of the light barriers is not possible as they are typically invisible in the camera images.

3.2 Pose Estimation for Robust Camera Calibration

We make some simplifying, but reasonable assumptions to use pose estimates to compute a solution for the image registration problem. Firstly, the image planes are approximately parallel to the flight path of the jumper and they are approximately aligned with the horizon. While we found both assumptions to be true in our setup, incorrect installation of the cameras could entail a rotation in the image plane. In this case, parameters for camera rotation have to be added to our model. Secondly, the detector used for pose estimation returns mostly decent pose estimates, although outliers should not distort the estimated model parameters. Finally, the flight path of the athlete between two camera views can be approximated with a second-degree polynomial.

Pose preprocessing. Under these assumptions, we first use a MobileNet [23] to detect a ski jumper in all images in every video. The pose of the jumper is estimated using a Convolutional Pose Machine [27] for each detection. We fine-tune both networks to our specific task to obtain better detections and pose estimates. The output of the pose estimator is a set of J joint coordinates for the head, shoulder, elbow, wrist, hip, knee and ankle of the body side facing the camera. The PCK [24] for the pose detector is depicted in Figure 2.

As discussed in the measurement setup in Section 3.1, the resulting pose estimates require some pre-processing. First, the scale of the jumper may be different for different camera views. Seconds, pose estimates are obtained from images with a sampling frequency of 50 fps, while force measurements are taken at 40 times this frequency, i.e., at 2 kHz. Third, we have to account for the fact that pose estimates may be noisy, mainly due to occlusion in the camera setup. Regarding the third statement, we found that the recall of our fine-tuned detector (Figure 2) is on a par with state-of-the-art pose estimation systems, if not even better due to the limited application domain. Within the generally accepted threshold of 0.2 of the upper body length of a person, we can detect >99% of all joints correctly. Even with a more restrictive threshold of 0.1, we can get a recall of 95%. In our application, this rate translates to a maximal deviation of only 5 pixels per joint, which is a very acceptable result. Nevertheless, we wish to account for this (potential) deviation and propose a method for robustly estimating the calibration parameters.

The first step is to account for different camera scales. For all poses obtained from one camera view C_i, we compute a scaling factor s_i based on the mean upper body size d_{upper} of the athlete, which is defined as the distance between shoulder and hip, in that shot:

$$s_i = \frac{d_{ref}}{d_{upper}}.$$

d_{ref} denotes a reference body size which is the same for all athletes in our database, i.e., all athlete have approximately the same upper body size in all shots. We rescale all pose estimates in the i-th camera view with s_i, resulting in the athlete having approximately the same relative size in all camera views. We found that the upper body size hardly changes within a camera view, can be detected very robustly (see Figure 2) and is, therefore, an excellent choice for obtaining a robust scale for each camera. To get a more robust estimate for the camera view scale, we exclude 10% of the largest and smallest detected upper body length when computing the mean in one view.

Overdetermined System. From our initial pose estimates of the ski jumper in all camera images of one run, we derive a robust polynomial regression for the trajectories of all joints of the athlete as follows. The location $l_j : \mathbb{R} \rightarrow \mathbb{R}^2$ of a joint j at any time t is given by a coordinate vector

$$l_j(t) = [\hat{x}_j(t), \hat{y}_j(t)]^T$$

where $\hat{x}_j(t)$ and $\hat{y}_j(t)$ are the two dimensions of the joint coordinate of joint j. A robust approximation of the joint trajectory and the camera offsets Δx and Δy for each location dimension is obtained using a 2^{nd} degree polynomial. For the x dimension, it is given by

$$\hat{x}_j(t) = \begin{cases} [\mathbf{t} \;\; 0]^T \cdot [\mathbf{b}_j \;\; \Delta x] & \text{if } l_j(t) \in C_i \\ [\mathbf{t} \;\; 1]^T \cdot [\mathbf{b}_j \;\; \Delta x] & \text{if } l_j(t) \in C_{i+1} \end{cases} \quad (1)$$

where $\mathbf{b}_j = [b_{j,2} \;\; b_{j,1} \;\; b_{j,0}]$ are the coefficients of the polynomial, Δx is the offset in x direction and $\mathbf{t} = [t^2 \;\; t \;\; 1]$ is a vector of polynomial variables (the times in the videos at which $\hat{x}_j(t)$ was predicted). In order to estimate the parameters $\mathbf{b}_j$ and camera offset Δx for two adjacent camera perspectives C_i and C_{i+1} using all J joints and all K detected poses from both views, we define an overdetermined system of equations

$$\hat{X} = AB. \quad (2)$$

Here, $\hat{X} \in \mathbb{R}^{KJ}$ is a vector of detected coordinates for each joint from each pose estimates. The parameter vector $B \in \mathbb{R}^{3J+1}$ is comprised of all polynomial coefficients and the offset, i.e,

$$B = [\mathbf{b}_1 \dots \mathbf{b}_J \quad \Delta x].$$

Each row in the sparse matrix $A \in \mathbb{R}^{KJ \times 3J+1}$ represents the polynomial variables for one joint detection (time the joint was detected) and is therefore connected to exactly one target value $\hat{x}_j(t) \in \hat{X}$. As the structure of this matrix is a bit intricate, we illustrate the composition of one row in A with a descriptive example. Assume that at time $t = 4$, we detected the shoulder joint ($j = 1$) at x position 356 in camera C_{i+1}. Then there would be a row $\mathbf{a}_i$ in matrix A coding this detection as

$$\mathbf{a}_i = [0 \;\; 0 \;\; 0 \;\; 16 \;\; 4 \;\; 1 \;\; 0 \dots 0 \;\; 1] = [0 \;\; 0 \;\; 0 \;\; t^2 \;\; t \;\; 1 \;\; 0 \dots 0 \;\; 1]$$

and the corresponding target value in $\hat{X}$ would be $\hat{X}_i = \hat{x}_1(4) = 356$. Note that according to Equation 2, this row of A is multiplied by the parameter vector B. Therefore, the zero values in this vector guarantee that all parameters that do not affect the estimation of the joint trajectory, i.e., all but $\mathbf{b}_1$ in this example, are neutralized. Furthermore, the last attribute in row vector $\mathbf{a}_i$ is set to 1 because this sample was taken from camera C_{i+1}. It would be $= 0$ if this specific joint were predicted in camera C_i. Thus, the estimated parameters express all trajectories relative to the reference camera C_i. While the above deviation was only made for $\hat{x}_j$, the approximation of $\hat{y}_j(t)$ is correspondingly defined according to Equation 1 and solved with a separate system.

Parameter Estimation. An optimal least-squares solution to Equation 2 is given by the Moore-Penrose inverse $B = (A^T A)^{-1} A^T \hat{X}$. However, as stated previously, we have to account for noise in the predicted poses. Hence, we choose to solve Equation 2 using a robust, iteratively reweighted least squares solution as presented in [4]:

$$B^{iter+1} = (A^T W^{(iter)} A)^{-1} A^T W^{(iter)} \hat{X}, \tag{3}$$

where W is a diagonal matrix of weights and (commonly) $W^{(0)} = I$. This algorithm determines a set of robust parameters B by iteratively reweighting the samples/rows in A with the inverse of their residual, computing a new weighted solution for the parameters B according to Equation 3. Thus, this iterative procedure assigns small weights to prediction outliers and thereby mitigates their influence on the final parameter set.

We apply this procedure to all pairs of adjacent camera views. The resulting trajectories and camera offsets for the two cameras pointed at the take-off table are visualized in Figure 1. The determined trajectory parameters are used to resample the joint positions to the sampling rate of the force plates. Thereby, one pose estimate can be assigned to each measured force value.

3.3 Synchronization of Force and Camera

With all camera images spatially registered, we are left with temporally matching camera views and measured forces. As stated previously, we know for each force value how far it was taken from the edge of the take-off table. Together with the upsampled pose estimates from the spatially registered camera views, temporal synchronization between both modalities is straightforward: The edge of the table is manually annotated in the respective view. The temporal synchronization anchor in the camera views is determined as the point where the angle of the ski jumper passes the edge and consequently leaves the last force plate. The synchronization point in the force-time series is merely the last point where a force measurement was taken on the take-off table. Both modalities are aligned at both temporal anchors. Finally, we crop the pose estimate time series and the synchronized force measurements to the same length of 1000 samples, equivalent to the last 25 frames of the jumper before take-off.

4 TEMPORAL CONVOLUTIONAL MODEL

We present a temporal convolutional sequence to sequence model based on the findings of Bai et al. in [1], who propose a basic architecture for a temporal convolutional network (TCN) which aims at combining the best recent practices in convolutional network design for sequence prediction. TCNs where originally designed to map a sequence of input length n to an output sequence of the same length using only causal convolutions, i.e., there is no "leakage" from future values to the past. TCNs aim to build a long effective history size, meaning that they are able to look very far back into the past to make a prediction. Commonly, TCNs are used for word-level language synthesis, machine translation modeling or polyphonic music modeling [6, 15, 25] and are tested on benchmark tasks defined on discrete valued input and output data. In our application, we modify the basic TCN blocks and propose a network for predicting a continuous time series from a continuous input.

4.1 Pose to Force Learning Task

Our task is to predict a time series representing the jump force of a ski jumper given a temporal sequence of continuously estimated athlete poses. The general nature of this task is to predict an output sequence of forces $f(0), ..., f(T)$ of length $T + 1$ from an input sequence of poses $(\hat{p}(0), ..., \hat{p}(T))$ given some convolutional deep neural network model $\mathcal{M} : P^{T+1} \rightarrow F^{T+1}$ that produces the mapping

$$f(0), ..., f(T) = \mathcal{M}(\hat{p}(0), ..., \hat{p}(T)).$$

A pose $\hat{p}(t) \in \mathbb{R}^{2J}$ is defined as an aggregation of J joint coordinates

$$\hat{p}(t) = [\hat{x}_1(t), \hat{y}_1(t), ..., \hat{x}_J(t), \hat{y}_J(t)]. \tag{4}$$

In terms of neural networks, the input signal P is a 1-dimensional array with $2 \cdot J$ channels, one for each resampled pose coordinate dimension.

4.2 Dilated Convolution Blocks

A major disadvantage of modeling a sequence to sequence network only with standard convolutional layers is that a very deep network with large kernel sizes has to be constructed in order to obtain a wide receptive field and thereby a large history size. Training very deep networks with large kernels from scratch is often not feasible, especially if comparatively few training data is available. Recently, dilated convolutions have been extensively researched for increasing the receptive field of a convolutional layer with a small kernel without having to add additional layers. A dilated convolution on an arbitrary input sequence $f \in \mathbb{R}^l$ of length l is convolved with a kernel $g \in \mathbb{R}^k$ of size $k << l$ using a dilation

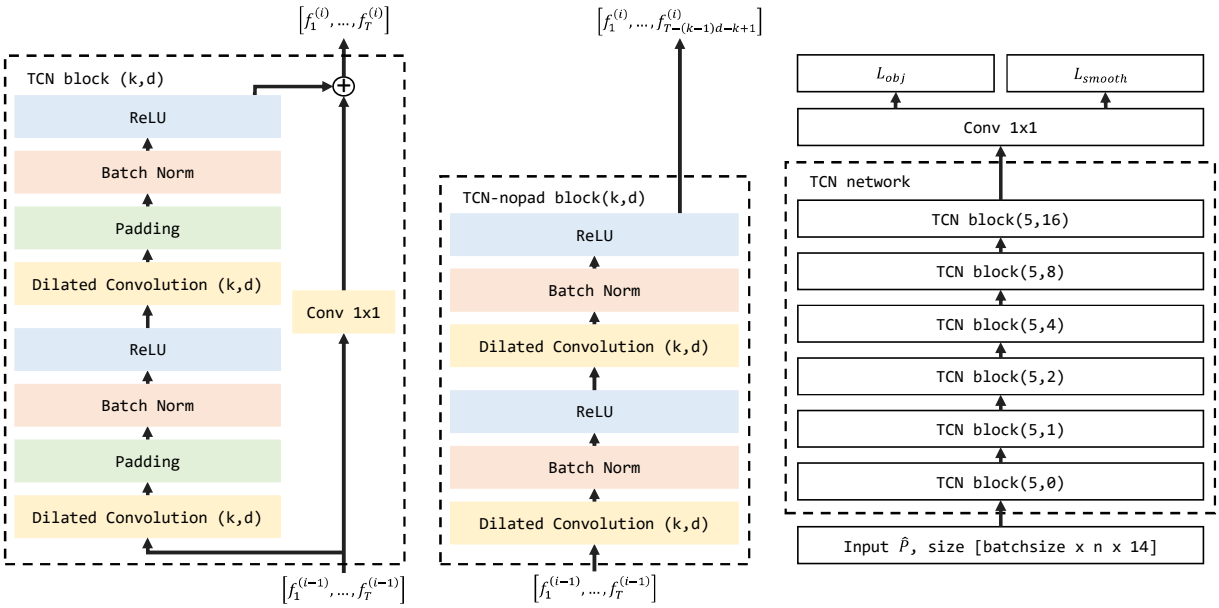

Figure 3: Left: standard TCN block inspired by [1]. Middle: TCN block without padding and residual connection. Right: An exemplary network architecture comprised of 6 TCN blocks (tcn_6).

factor d through

$$f *_d g[t] = \sum_{i=0}^{l-1} f[t - id] \cdot g[i] \tag{5}$$

Dilated convolution spreads the filter coefficients over a larger area of the input signal, increasing the receptive field and thereby - in a temporal context - expands the history of past values.

Inspired by [1], we define a temporal convolution network block (TCN block) as the basic building block. Such a block is depicted in Figure 3. This block includes two consecutive dilated convolutional layers with additional padding, augmenting the output features to the same size as the input features. Each convolutional layer is followed by batch-normalization [13] and a Rectified Linear Unit [21]. A residual connection [11] is added to the output of the block, which allows for learning only the difference between the input and the output of the block. Each TCN block has two parameters k and d for the kernel and dilation size, respectively.

In the standard definition of the TCN block, the length of the input signal does not change throughout the network due to extensive padding. However, we found that there is a drawback to dilated convolutions which arises especially when the dilation factors get very large. In this case, the effective size of the kernel is a huge and the amount of zero-padding has to be equally increased to ensure that the output signal has the same length as the input signal. We found that for large dilation factors (i.e., $d > 10$ at kernel size $k = 5$), extensive padding introduces a lot of noise at the edges of the final output signal (see Figure 4). Consequently, the gradients produced from noise at the edges of the output signal may introduce a spurious training stimulus for all underlying convolutional kernels.

To examine this effect, we define a variation of TCN block which is also depicted in Figure 3. This block waives the padding operation after each convolutional layer. Consequently, the skip connection has to be dropped due to the difference in input and output size of the block. We found that this is not a disadvantage to the performance of the network, as residual connections are inherently beneficial in very deep networks, while our overall network depth does not exceed 13 convolutional layers. We refer to a TCN block without padding as tcn_nopad. The output shapes of the last TCN-output layer are listed in Section 5.

4.3 Network Structure and Losses

The overall structure of a convolutional sequence to sequence network is built on TCN blocks (tcn or tcn_nopad). A sample architecture is depicted in Figure 3. The input of the network is a time series of resampled pose estimates as described in Section 3.2. The input layer is followed by some TCN blocks. In order to increase the receptive field of the network quickly without having to add too many blocks, the dilation factor for block q in the sequence is set to be $d = 2^q$. In our experiments, the depth of the network does not exceed 7 TCN blocks. We additionally experiment with different quantities of kernels per block and compare classical design approaches based on a steadily increasing number of kernels with strategies where the amount of kernel first increases and then decreases. Details of different network parameters are listed in Table 1. A final convolutional layer terminates the network with kernel size 1 and linear activation, which reduces the output of the last layer to a one-dimensional sequence of force values.

The training loss of our convolutional sequence to sequence networks is defined by three partial losses. First, the objective loss

name	len. input	len. output	# tcn blocks	# channels/block	kernelsize/layer
tcn_6	950	950	6	[32,64,128,64,32,16]	5
tcn_5	950	950	5	[32,64,128,64,32]	5
tcn_4	950	950	4	[32,64,128,64]	5
tcn_nopad_6	950	446	6	[32,64,128,64,32]	5
tcn_nopad_5	950	702	5	[32,64,128,64,32,16]	5
tcn_nopad_4	950	830	5	[32,64,128,64,32,16]	5
tcn_6_wide	950	950	6	[32,64,128,128,128,128]	5
tcn_5_wide	950	950	5	[32,64,128,128,128]	5
tcn_7	950	950	7	[32,64,128,128,64,32,16]	5
tcn_2_shallow	950	950	2	[64,128] w/ dilation [2,16]	5

Table 1: Overview over different tested network architectures. *tcn_x* networks are build from standard tcn blocks, *tcn_nopad_x* nets discard padding. *tcn_x_wide* architectures increase the number of kernels from layer to layer.

between the ground truth force F and the predicted force $\bar{F}$ is defined by the piece-wise Huber-Loss with $\delta = 1$, defined as

$$L_{obj}(F - \bar{F}, \delta) = \begin{cases} 0.5(F - \bar{F})^2 & |F - \bar{F}| \leq \delta \\ \delta(|F - \bar{F}| - 0.5\delta) & \text{otherwise.} \end{cases}$$

Second, in order to avoid overfitting convolutional layers, a L_2-regularization loss per layer is introduced. All regularization losses are aggregated in the overall regularization loss L_{reg}. Third, we found that the output of the network is often noisy. In order to encourage smoothness in the final force predictions, we define a smoothness loss $L_{smooth} = L_{obj}(\frac{d}{dt}F - \frac{d}{dt}\bar{F}, \delta)$ to be the Huber-Loss of the difference between derivatives of ground-truth and prediction. The derivative of either time series is simply determined numerically by $\frac{d}{dt}F(t) = F(t) - F(t-1)$.

The total network loss is defined as the weighted sum of all partial losses:

$$L_{total} = L_{obj} + \alpha L_{reg} + \beta L_{smooth}. \quad (6)$$

5 EXPERIMENTS

The prediction of continuous dynamic parameters of ski jumpers (and athletes in general) solely from continuous pose estimates using convolutional sequence models is a novel problem in the scientific community with no published results yet. Hence, we focus on analyzing different network structures quantitatively and qualitatively.

5.1 Dataset

We build a dataset from 225 videos of different ski jumpers for training and testing network architectures mentioned above. Each video depicts all consecutive camera perspectives for one single jump. A shot detection algorithm is applied to extract the original camera views. The videos are recorded at 25 interlaced frames per second, leaving us with 50 full frames after de-interlacing. The footage covers a wide range of recording scenarios (day vs. night, good vs. bad vs. foggy weather, snowy and rainy scenes, winter and summer footage) and a variety of ski jumpers of all ages and statures. As described in Section 3.2, we estimate all poses in each video and derive calibration and synchronization parameters. The dataset is split into 200 videos for training and 25 for testing. We artificially augment the size of both sets by cropping sequences of length 950 from the original data with length 1000, which increases the dataset size by a factor of 10. During training, all sequences are randomly shuffled.

While the pose input of the network was normalized during the calibration procedure, we have not yet talked about the output. If not stated otherwise, all force series are normalized by the largest occurring force value in the dataset, which maps all forces into the interval $[0, 1]$. For tcn_nopad networks, we resample the force time series to the proper network output size using FFT-resampling [14].

5.2 Error measures

We report the Mean Squared error (MSE) and median of the squared errors between the ground truth $F(t)$ and the predicted forces $\bar{F}(t)$ for different network structures. All errors are computed over all test-videos and all force values (see Table 2 and Figure 5).

PCKF Recall. In order to give the reader a better intuition of network performances, we additionally try to answer a real-world question asked by coaches during the evaluation process: At which time in the video is the maximal take-off force observed? In a training situation, the corresponding frame gives the athlete a visual clue for adjusting the jumping motion in the next run.

We take the argument of the maximal force value $argmax(\bar{F}(t))$ and the corresponding pose estimate, which depicts the athlete's posture during maximal leg pressure. The temporal occurrence of this pose is compared to the maximal force occurrence frame of the ground-truth. For measuring the recall of this comparison, we take inspiration from the percentage-of-correct-key-points measure (PCK, [24]), which counts detection as correct if the distance of a predicted key-point from the ground truth is below a (variable) threshold τ. We define a similar measure for key-frame detection, denoted as percentage-of-correct-key-frames (PCKF), which counts a predicted maximal force occurrence as correct if the temporal distance to the ground-truth frame falls within a threshold τ. In case of unpadded networks (tcn_unpad), where the output sequence length is implicitly defined by the number of dilated convolutional layers (without padding) and the filter size, we scale the offset between prediction and ground-truth accordingly.

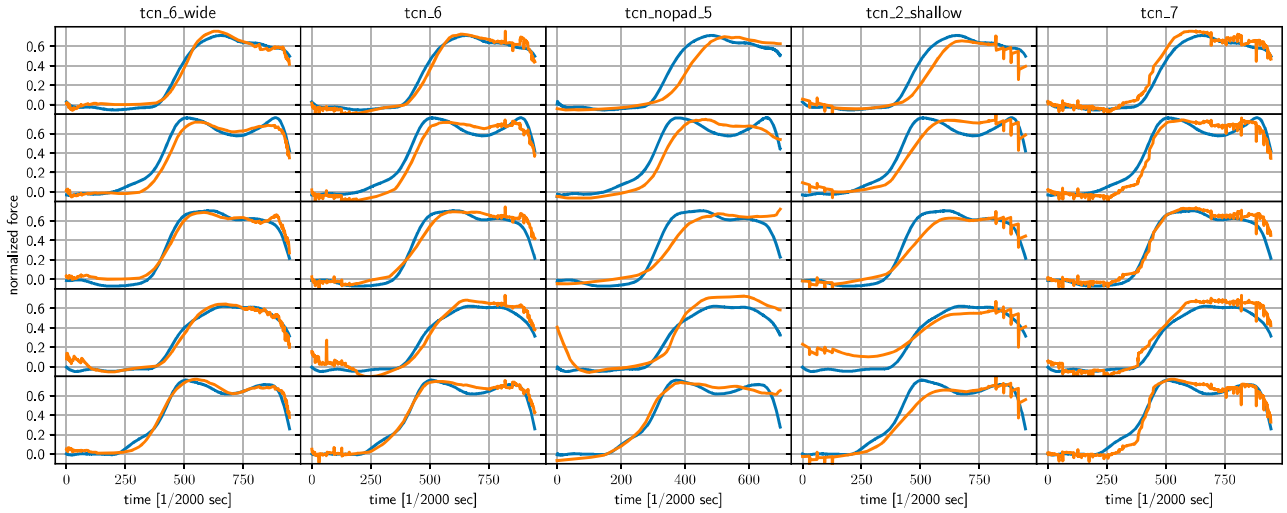

Figure 4: Qualitative outputs of different architectures. The best performing networks in terms of MSE (first three columns) and two more exotic variants (last two columns) are compared for five differnet force series (rows). Ground-truth measurements are depicted in blue, predictions in orange.

5.3 Tested network structures and Training parameters

With the novelty of the discussed application, specifically in the context of convolutional sequence to sequence networks, a state-of-the-art for network architecture and parameters is widely unknown. We use this opportunity to perform our experiments on a variety of networks (Table 1). Three different network types are examined. The basic tcn-network is implemented with 4 to 6 tcn blocks (tcn_4, tcn_5 and tcn_6) and defines the baseline of our experiments. Non-padded versions (tcn_unpad_4, tcn_unpad_5, tcn_unpad_4) are used to examine the influence of noise in output sequences. All networks use a bicone shaped number of filters throughout the layers. We compare this parameter set to two networks tcn_5_wide and tcn_6_wide, which implement a more traditional distribution of kernels, i.e., the number of kernels is steadily increased throughout the network. We also experiment with two more exotic networks, namely tcn_7 and tcn_2_shallow. tcn_7 is a 15 layer network, i.e., 7 TCN blocks and the final layer, where the receptive field covers the entire pose input of the network. tcn_2_shallow is a shallow net comprised of 2 TCN blocks. The dilation factors for both blocks were set to 2 and 16, respectively. With this network, we want to examine if the reconstruction of a force signal can be achieved with very few layers with large dilation factors. An overview of all network parameters is given in Table 1.

The layers in all networks are initialized with variance scaling initialization [10] and trained until convergence in mini-batches of size 16 with a learning rate of 0.01 and a learning rate reduction of 0.1 every 3 epochs. The loss weights for regularization loss was set to $\alpha = 0.0001$, while the smoothness loss was set to $\beta = 1$

Network	MSE	median
tcn_6	**0.071**	0.049
tcn_5	0.074	0.051
tcn_4	0.088	0.060
tcn_nopad_6	0.105	0.067
tcn_nopad_5	0.087	0.057
tcn_nopad_4	0.091	0.061
tcn_6_wide	**0.071**	0.049
tcn_5_wide	0.073	**0.047**
tcn_7	**0.062**	**0.043**
tcn_2_shallow	0.093	0.065

Table 2: MSE and median of squared errors between ground truth and predicted forces.

5.4 Results

Before presenting the results for tcn_7 and tcn_2_shallow, we first discuss the basic architectures.

MSE and prediction quality. The results of our experiments are listed in Table 2 and, for the reader's convenience, are visualized in Figure 5. We first find that dropping the padding does not entail good predictions regarding MSE. All architectures without padding performed considerably worse in all experiments. A visual inspection of some outputs in Figure 4 (middle column) supports the quantitative errors. All tcn networks and their tcn_wide variants seem to perform much better. MSE errors indicate that more layers in a network result in superior performance over networks with a smaller receptive field. Comparing the quality of the best performing networks tcn_6 and tcn_6_wide (left two columns in Figure 4), we see that tcn_6 produces more noise at the edge of the signals, despite them having the same MSE and median error. In

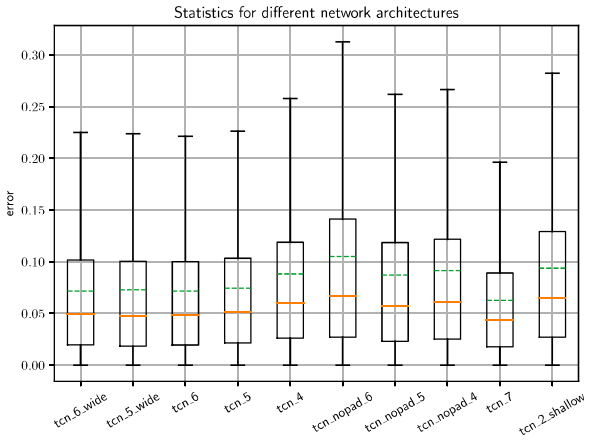

Figure 5: Network architecture error statistics. Whiskers indicate 2nd and 98th percentile, orange lines the median error, green lines the mean error. Outliers are omitted.

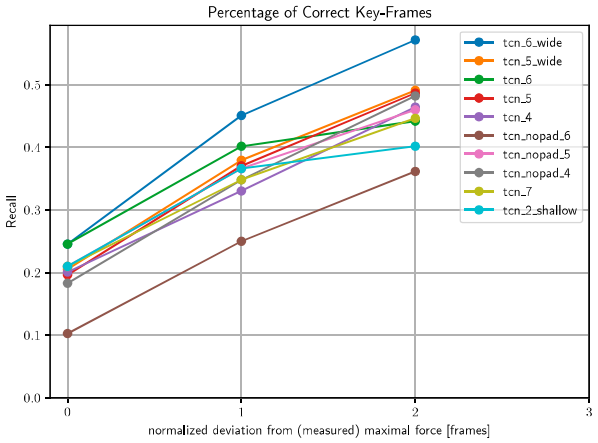

Figure 6: Percentage of correctly identified key-frames depicting the moment of the maximal leg pressure before the jump. The best performing network predicts 25% of the frames perfectly correct and 55% within ± 2 frames.

general, the observed noise is a product of the large dilation factors and stems from the convolution of a filter with large amounts of padded zero-values. The prediction noise can be observed in all padded tcn networks. From our experiments, we conclude that adding more filters to the networks, especially in later layers, might lower this noise.

The evidence for the hypothesis that deeper networks are superior in reproducing the force signal is supported by the error of tcn_2_shallow, which performs similarly to the tcn_nopad versions. While this network grasps the general trend of a force graph (4th column in Figure 4), a significant amount of raw noise can be observed at the edge of the predicted signals, despite a large number of filters in this model.

The best results are obtained by the deepest network tcn_7. As previously stated, the perceptive field of this network covers the complete input. While it produces a relatively small MSE, we can also observe in column 5 of Figure 4 that it creates more noise in the prediction than all other architectures. While the overall performance of tcn_7 may be favorable, the noise problem has to be addressed to deliver a visually superior prediction.

PCKF. We conclude our experimental section with a short discussion of the PCKF performance of all models, depicted in Figure 6. Our best models tcn_6 and tcn_6_wide detect 25% of the sought-after frames correctly. Within a deviation of 2 frames, tcn_6_wide - despite producing the same MSE as tcn_6 - surprisingly produces a superior recall of 55%. Most of the models perform surprisingly well, although no model beats tcn_6_wide. It may also be surprising that the best model regarding MSE, tcn_7, does not even come close to the best scores. This may be attributed to the fact that we intentionally did not perform any post-processing on the predicted force series, at which point the noise in the tcn_7 output is detrimental.

On the question of why the recall of the networks is comparatively small in absolute terms, we point to Figure 4. We found that force measurements very often have multiple local maxima, sometimes very close together, often with very similar absolute values. This makes the identification of force peaks a great challenge in general, which we will accept in the future.

6 CONCLUSION AND FUTURE WORK

In this paper, we discussed convolutional sequence to sequence networks for the multimodal prediction of jump forces of ski jumpers. The prediction networks use the estimated pose of the jumper to infer a series of force measurements, recorded with force plates at training time. The synchronization between all modalities, i.e., multiple cameras and multiple force measurement plates, was implemented through least squares optimization for the joint trajectories of the athlete. We finally present purely convolutional network structures and demonstrate their effectiveness experimentally.

While the experimental results are very promising, problems like prediction noise have to be solved to produce more competitive force estimates in the future. Furthermore, an interesting question arises when the problem statement is reversed: Is it possible to predict the pose of a ski jumper with a convolutional sequence to sequence model, given only the measured force values of a jump? This question and many others remain future work.

ACKNOWLEDGMENTS

This work was funded by the Federal Institute for Sports Science based on a resolution of the German Bundestag. We would like to thank the Institute for Applied Training Science (IAT) Leipzig, especially Dr. Sören Müller and Dr. Sascha Kreibich, for collecting and providing the data.

REFERENCES

[1] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *CoRR* abs/1803.01271 (2018). arXiv:1803.01271 http://arxiv.org/abs/1803.01271

[2] Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5, 2 (Mar 1994), 157–166. https://doi.org/10.1109/72.279181

[3] M.A. Brubaker, L. Sigal, and D.J. Fleet. 2009. Estimating Contact Dynamics. In *Proc. IEEE International Conference in Computer Vision (ICCV)*.

[4] C. S. Burrus, J. A. Barreto, and I. W. Selesnick. 1994. Iterative reweighted least-squares design of FIR filters. *IEEE Transactions on Signal Processing* 42, 11 (Nov 1994), 2926–2936.

[5] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Association for Computational Linguistics, 103–111. https://doi.org/10.3115/v1/W14-4012

[6] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language Modeling with Gated Convolutional Networks. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Doina Precup and Yee Whye Teh (Eds.), Vol. 70. PMLR, International Convention Centre, Sydney, Australia, 933–941. http://proceedings.mlr.press/v70/dauphin17a.html

[7] Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. 2015. Learning Visual Predictive Models of Physics for Playing Billiards. *CoRR* abs/1511.07404 (2015). arXiv:1511.07404 http://arxiv.org/abs/1511.07404

[8] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. 2015. Recurrent Network Models for Human Dynamics. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV) (ICCV '15)*. IEEE Computer Society, Washington, DC, USA, 4346–4354. https://doi.org/10.1109/ICCV.2015.494

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *ICML*.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV) (ICCV '15)*. IEEE Computer Society, Washington, DC, USA, 1026–1034. https://doi.org/10.1109/ICCV.2015.123

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 770–778.

[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[13] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. 448–456. http://jmlr.org/proceedings/papers/v37/ioffe15.pdf

[14] Eric Jones, Travis Oliphant, Pearu Peterson, et al. 2001–. SciPy: Open source scientific tools for Python. http://www.scipy.org/

[15] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural Machine Translation in Linear Time. *CoRR* abs/1610.10099 (2016).

[16] K. Lee and R.D. Green. 2005. Temporally Synchronizing Image Sequences Using Motion Kinematics. In *Proc. Image and Vision Computing*.

[17] Chen Li, Zhen Zhang, Wee Sun Lee, and Gim Hee Lee. 2018. Convolutional Sequence to Sequence Model for Human Dynamics. *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* abs/1805.00655 (2018).

[18] Julieta Martinez, Michael J. Black, and Javier Romero. 2017. On human motion prediction using recurrent neural networks. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017*. IEEE, Piscataway, NJ, USA.

[19] Roozbeh Mottaghi, Hessam Bagherinezhad, Mohammad Rastegari, and Ali Farhadi. 2015. Newtonian Image Understanding: Unfolding the Dynamics of Objects in Static Images. *CoRR* abs/1511.04048 (2015). arXiv:1511.04048 http://arxiv.org/abs/1511.04048

[20] Roozbeh Mottaghi, Mohammad Rastegari, Abhinav Gupta, and Ali Farhadi. 2016. "What happens if..." Learning to Predict the Effect of Forces in Images. *CoRR* abs/1603.05600 (2016). arXiv:1603.05600 http://arxiv.org/abs/1603.05600

[21] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML'10)*. Omnipress, USA, 807–814. http://dl.acm.org/citation.cfm?id=3104322.3104425

[22] F. Padua, R. Carceroni, G. Santos, and K. Kutulakos. 2010. Linear Sequence-to-Sequence Alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 2 (Feb 2010), 304–320. https://doi.org/10.1109/TPAMI.2008.301

[23] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation. *CoRR* abs/1801.04381 (2018). arXiv:1801.04381 http://arxiv.org/abs/1801.04381

[24] Benjamin Sapp and Ben Taskar. 2013. MODEC: Multimodal Decomposable Models for Human Pose Estimation. In *CVPR*. IEEE Computer Society, 3674–3681.

[25] AÃđron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. In *Arxiv*. https://arxiv.org/abs/1609.03499

[26] D. Wedge, P. Kovesi, and Du Huynh. 2005. Trajectory Based Video Sequence Synchronization. In *Digital Image Computing: Techniques and Applications (DICTA'05)*. 13–13. https://doi.org/10.1109/DICTA.2005.82

[27] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. 2016. Convolutional pose machines. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[28] Jiajun Wu, Ilker Yildirim, Joseph J Lim, Bill Freeman, and Josh Tenenbaum. 2015. Galileo: Perceiving Physical Object Properties by Integrating a Physics Engine with Deep Learning. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.). Curran Associates, Inc., 127–135.