Reconstructing Spatial Aspects of Motion by Image-to-Path Deep Neural Networks

Rok Pahič[1,2], Andrej Gams[1], Aleš Ude[1,3]

***Abstract*—The choice of an appropriate representation is important when reconstructing motion from images. In this paper we propose a new type of deep neural network (DNN) that maps images to spatial paths represented by a recently introduced motion representation called arc-length dynamic movement primitive (AL-DMP). This representation separates the spatial from temporal aspects of motion and is therefore suitable for processing data that do not contain temporal information. We propose a physically meaningful loss function for training of AL-DMPs, which improves the performance of the trained DNN, and derive its gradients, which are needed to apply the backpropagation algorithm. Moreover, the proposed DNN architecture supports the processing of variable-size input images and images with cluttered background.**

The developed approach was applied to the reproduction of handwritten digits from single images that do not contain temporal aspects of motion. Our experiments also show that the proposed network can be applied to input images of sizes that are different from the size of training images. Finally, the proposed approach was successfully applied for reproducing real handwritten digits with a humanoid robot.

***Index Terms*—Perception-action coupling, novel deep learning methods, deep learning for visual perception**

I. INTRODUCTION

DEEP neural networks (DNNs) have proven to be effective at improving the capabilities of perceptual systems in many different application areas [1]. In robotics, DNNs can provide the functionality needed to learn nonlinear mappings between perception and action. Such mappings are the key to the development of autonomous robots that need to operate in real unstructured environments. By coupling perception and action, a robot can form higher-level concepts such as object-action complexes [2], which have been proposed to bind objects, actions, and the accompanying attributes in a causal model. Together with suitable representations, DNNs can contribute to the development of a new generation of cognitive robots.

In our previous work [3], we successfully coupled perception and action by applying DNNs to compute dynamic movement primitives (DMPs) [4], [5] from perceptual data. The proposed approach was employed for the reproduction of handwriting motion from raw images of digits. A distinguishing advantage of the developed methodology is the definition of an appropriate evaluation metric, which can be used to train DNNs that output DMP parameters.

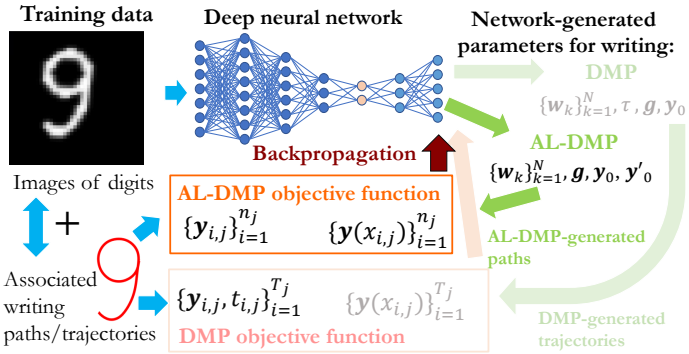


Fig. 1: Comparison of training with AL-DMPs (bold) and DMPs (shaded) parameters. An input image is fed to a DNN, which transforms the image into AL-DMP (DMP) parameters. The output AL-DMP (DMP) is then used to generate a sequence of points along the path (trajectory). During training, the generated points are compared with the data points along the training path (as a function of arc-length) / trajectory (as a function of time). The parameters of DNN are estimated by backpropagation.

However, images of handwritten digits do not contain information about the temporal course of handwriting movement. The same digit can be written at different speeds, but the approach from [3] cannot deal with data containing digits written at multiple speeds. Here we address this problem by incorporating a suitable representation that does not require information about the temporal course of motion, i. e., the *arc-length dynamic movement primitive* (AL-DMP) representation [6], [7]. The process of training DNNs to map raw images to DMPs or AL-DMPs is graphically shown in Fig. 1. Another problem not addressed in [3] is how to deal with images of different sizes and containing cluttered backgrounds. Both issues are addressed in this paper.

As it is standard in robotics, in the following we refer to *path* when given a set of points that lead a robot from the starting point to the final point without specifying how quickly the robot should move. We refer to *trajectory* if the robot is given a path together with a temporal schedule for how to get from the initial to the final point.

A. Related Work

In this paper we propose to make use of AL-DMP representation instead of standard DMP representation when the

Manuscript received: July 17, 2020; Accepted November 16, 2020. This letter was recommended for publication by Associate Editor A. Banerjee and Editor Dan Popa upon evaluation of the Reviewers' comments. This work was supported by the program group "Automation, robotics, and biocybernetics", P2-0076, funded by the Slovenian Research Agency.

[1]Humanoid and Cognitive Robotics Lab, Dept. of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Ljubljana, Slovenia

[2]Jožef Stefan International Postgraduate School, Ljubljana, Slovenia

[3]Faculty of Electrical Engineering, University of Ljubljana, Slovenia

Digital Object Identifier (DOI): 10.1109/LRA.2020.3039937.

temporal information is not available. The AL-DMP representation is advantageous in this case because it effectively separates the spatial from temporal component of motion. It can therefore be used even when the timing of motion is not available or is different due to the varying speed of motion. Both DMPs and AL-DMPs have been extensively used in the scope of learning by demonstration. Other representations have also been utilized, e. g., polynomial splines, radial basis functions, Gaussian Mixture Models, probabilistic movement primitives, etc. See [8] for a review. However, DMPs and AL-DMPs have many advantages because they can be modulated and are robust to perturbations.

Standard DNNs require inputs of constant size and therefore cannot process arbitrary sized input images. This can be addressed in several ways, e. g. by spatial pyramid pooling [9] or by a global average pooling layer [10], which takes an average over each feature map in the last convolutional layer to normalize the size. Another possibility is to use the region of interest pooling as in [11]. In each region, the max pooling is used to create a fixed size vector.

In contrast to [10], in this paper we propose a global max pooling layer, which takes the maximum value from each feature map in the last convolutional layer instead of the average. The resulting fixed-size vector forms the latent layer, which is used as input to the subsequent decoder layers. These layers apply regression to compute the final output values. Thus not only does our approach deal with regression as opposed to classification, but the use of the global pooling layer also allows the DNN to filter out background noise and focus on the most relevant local features. In our experiments we show that the proposed DNN architecture is capable of being trained with digit images of a particular size and then generalizes to process images of an arbitrary size.

II. Normalized Arc Length Dynamic Movement Primitives

As explained above, dynamic movement primitives (DMPs) [4] are often used to represent trajectories. A DMP is defined as a second-order linear dynamical system with an added nonlinear forcing term that encodes a time-dependent movement trajectory $\boldsymbol{y}(t) \in \mathbb{R}^d$, where d is the dimension of the space in which robot motion is specified (robot joint angles or Cartesian space).

The DMP movement representation has many advantages for robot trajectory generation and control. It encodes both spatial and temporal aspects of motion. However, it is not easy to separate the two aspects in the DMP representation. For this reason, Gašpar et al. [7] reformulated the DMP movement representation based on the arc length parameterization of spatial paths. For a time-parameterized trajectory $\mathbf{y}(t)$, the spatial length of the trajectory (also called arc length) up to time t can be computed as

$$s(t) = \int_0^t \|\dot{\mathbf{y}}(u)\| du. \tag{1}$$

Given the duration T of the complete movement, the total length L of the spatial path traversed by the trajectory is given by

$$L = s(T) = \int_0^T \|\dot{\mathbf{y}}(u)\| du, \tag{2}$$

The movement speed is related to the temporal evolution of arc-length and can be calculated as the time derivative of s

$$\dot{s}(t) = \|\dot{\mathbf{y}}(t)\|. \tag{3}$$

In any valid re-parameterization $\boldsymbol{y}(s(t))$ of the time dependent trajectory $\boldsymbol{y}(t)$, the parameter s needs to be strictly increasing, i. e. $\dot{s}(t) > 0, \forall t$ [12]. Otherwise it is not possible to compute the derivatives of the path $\boldsymbol{y}(s)$ with respect to the parameter s. According to Eq. (3), the derivative of speed always fulfils condition $\dot{s}(t) \geq 0$. If there exist times t where the speed of motion equals zero, the trajectory needs to be subdivided into segments of non-zero speed, except possibly at both ends of the trajectory. We can then use the arc length for re-paramaterization of each trajectory segment. Note that the derivatives of the path with respect to arc length have unit norm, i.e. $\|\boldsymbol{y}'(s)\| = 1$. This can be easily proven by computing the derivative of $\boldsymbol{y}(s(t))$ with respect to s and taking into account Eq. (3)

$$\|\boldsymbol{y}'\| = \left\| \frac{d\boldsymbol{y}}{ds} \right\| = \left\| \frac{d\boldsymbol{y}/dt}{ds/dt} \right\| = \frac{\|\dot{\boldsymbol{y}}\|}{\|\dot{\boldsymbol{y}}\|} = 1. \tag{4}$$

Based on the DMP representation and arc length parametrization, Gašpar et al. [7] proposed a speed-independent differential equation system to encode the desired path for robot motion. In the proposed representation, the derivatives are computed with respect to arc length s instead of time t as in standard DMPs. The following system of differential equations was proposed:

$$L\boldsymbol{z}' = \alpha_z(\beta_z(\boldsymbol{g} - \boldsymbol{y}) - \boldsymbol{z}) + \mathbf{F}(x), \tag{5}$$

$$L\boldsymbol{y}' = \boldsymbol{z}, \tag{6}$$

$$Lx' = -\alpha_x x. \tag{7}$$

Here $\boldsymbol{g} \in \mathbb{R}^d$ is the final position on the path and $\mathbf{F}(x) \in \mathbb{R}^d$ a nonlinear forcing term. All derivatives are taken with respect to arc-length and are denoted by $'$. Eq. (5) – (7) define the arc length dynamic movement primitive (AL-DMP).

The forcing term $\mathbf{F}(x)$ is usually defined as a linear combination of radial basis functions

$$\mathbf{F}(x) = \frac{\sum_{k=1}^N \boldsymbol{w}_k \Psi_k(x)}{\sum_{k=1}^N \Psi_k(x)} x, \tag{8}$$

$$\Psi_k(x) = \exp\left(-h_k \left(x - c_k\right)^2\right), \tag{9}$$

where c_k are the centers of Gaussians distributed along the phase of the path, and h_k their widths. The role of weights $\boldsymbol{w}_k \in \mathbb{R}^d$ in $\mathbf{F}(x)$ is to adapt the dynamics of (5) – (6) to the desired path $\boldsymbol{y}(s)$, thus enabling the system to reproduce any smooth movement from the initial position $\boldsymbol{y}_0$ to the final configuration $\boldsymbol{g}$.

The parameters α_z, β_z, and α_x are usually constant and do not change when encoding different paths. Thus to fully specify an AL-DMP, the following parameters need to be provided: $\{\boldsymbol{w}_k\}_{k=1}^N$, $\boldsymbol{g}$, L, $\boldsymbol{y}_0$, and $\boldsymbol{y}_0'$. While $\boldsymbol{y}_0$ and $\boldsymbol{y}_0'$ do not appear in the equation system (5) – (7), they are needed

to initialize the integration of AL-DMP. The reason for this is that unlike in standard DMPs, where the starting velocity is usually set to zero, this is not the case with AL-DMPs because according to Eq. (4), the arc length derivative of $\boldsymbol{y}$ is a unity vector. Thus $\boldsymbol{y}'$ is always different from zero and must be provided to initialize the integration.

Note that among $\{\boldsymbol{w}_k\}_{k=1}^N$, $\boldsymbol{g}$, L, $\boldsymbol{y}_0$, and $\boldsymbol{y}'_0$, L is the only parameter in the differential equation system (5) – (7) that affects all dimensions of the control variable $\boldsymbol{y}$. We therefore reformulate system (5) – (7) to avoid using L. This can be achieved by re-parameterization with normalized arc length, i.e. $\widetilde{s} = s/L$, $ds/d\widetilde{s} = L$. Let's define $\widetilde{\boldsymbol{y}}(\widetilde{s}) = \boldsymbol{y}(\widetilde{s}L) = \boldsymbol{y}(s)$, $\widetilde{\boldsymbol{z}}(\widetilde{s}) = \boldsymbol{z}(s)$, and $\widetilde{x}(\widetilde{s}) = x(s)$. This re-parameterization changes the spatial derivatives as follows:

$$\widetilde{\boldsymbol{y}}' = \frac{d\widetilde{\boldsymbol{y}}}{d\widetilde{s}} = \frac{d\boldsymbol{y}}{ds}\frac{ds}{d\widetilde{s}} = L\boldsymbol{y}'. \tag{10}$$

Similarly we obtain

$$\widetilde{\boldsymbol{z}}' = L\boldsymbol{z},\ \widetilde{x}' = Lx'. \tag{11}$$

Thus we can rewrite differential equation system (5) – (7)

$$\widetilde{\boldsymbol{z}}' = \alpha_z(\beta_z(\boldsymbol{g} - \widetilde{\boldsymbol{y}}) - \widetilde{\boldsymbol{z}}) + \mathbf{F}(\widetilde{x}), \tag{12}$$

$$\widetilde{\boldsymbol{y}}' = \widetilde{\boldsymbol{z}}, \tag{13}$$

$$\widetilde{x}' = -\alpha_x\widetilde{x}. \tag{14}$$

Note that the above equation system does not contain L. We will see later in the paper that such a reformulation leads to simplified calculations of gradients of criterion functions. The consequence of normalization is also that the norm of velocity with respect to arc length is no longer unity vector, but has norm equal to the arc length L

$$\|\widetilde{\boldsymbol{y}}'\| = \left\|\frac{d\widetilde{\boldsymbol{y}}}{d\widetilde{s}}\right\| = \left\|\frac{d\boldsymbol{y}}{ds}\frac{ds}{d\widetilde{s}}\right\| = L\,\|\boldsymbol{y}'\| = L. \tag{15}$$

For the sake of clarity we omit in the rest of this paper the modifier $\widetilde{\ }$ when referring to differential equation system (12) – (14) and its parameters. We call the resulting representation a *normalized arc length dynamic movement primitive*.

A. Estimation of normalized AL-DMPs

Given a sequence of points $\{\boldsymbol{y}_i\}_{i=0}^n,\ \boldsymbol{y}_i \neq \boldsymbol{y}_{i-1}\ \forall i$, on the robot path, we can estimate the arc length L as follows

$$L = \sum_{i=1}^{n} \|\boldsymbol{y}_i - \boldsymbol{y}_{i-1}\|. \tag{16}$$

Next we compute the normalized arc length parameters s_i

$$s_i = s_{i-1} + \frac{1}{L}\|\boldsymbol{y}_i - \boldsymbol{y}_{i-1}\|, i = 1, \ldots, n,\ s_0 = 0. \tag{17}$$

Given the fact that according to Eq. (15) $\|\boldsymbol{y}'\| = L$, we can approximate the normalized arc length derivatives using the following formulas for all i, $i = 0, \ldots, n-1$:

$$\boldsymbol{y}'_i = L\frac{\boldsymbol{y}_{i+1} - \boldsymbol{y}_i}{\|\boldsymbol{y}_{i+1} - \boldsymbol{y}_i\|}, \tag{18}$$

$$\boldsymbol{y}''_i = \frac{\boldsymbol{y}'_{i+1} - \boldsymbol{y}'_i}{s_{i+1} - s_i}. \tag{19}$$

From these data we can copy the initial position and velocity $\boldsymbol{y}_0$ and $\boldsymbol{y}'_0$ as well as the goal $\boldsymbol{g} = \boldsymbol{y}_n$. The weights $\boldsymbol{w}_k$ of the forcing term (8) can be estimated by rewriting Eq. (5) – (6) as a second order differential equation for all i, $i = 0, \ldots, n$:

$$\boldsymbol{y}''_i - \alpha_z(\beta_z(\boldsymbol{g} - \boldsymbol{y}_i) - \boldsymbol{y}'_i) = \frac{\sum_{k=1}^N \boldsymbol{w}_k\Psi_k(x_i)}{\sum_{k=1}^N \Psi_k(x_i)}x_i. \tag{20}$$

In Eq. (20), the weights $\boldsymbol{w}_k$ appear linearly, thus given $\boldsymbol{y}_i$, $\boldsymbol{y}'_i$, $\boldsymbol{y}''_i$, and x_i, $i = 0, \ldots, n$, they can be computed by solving a linear least squares optimization problem.

III. Training of deep neural networks with normalized AL-DMPs at the output

Our goal is to develop a DNN that transforms raw images at the input to robot paths at the output. In [3] we showed how this can be accomplished when mapping images to DMPs. In the following we show how such a mapping can be computed for normalized AL-DMPs.

We start by specifying training data as pairs of images and the associated handwriting paths:

$$\mathbf{D} = \{\mathbf{C}_j, \mathbf{M}_j\}_{j=1}^P, \tag{21}$$

where P is the number of training pairs, $\mathbf{C}_j \in \mathbb{R}^{H\times W}$ are the input images of width W and height H, and $\mathbf{M}_j$ the corresponding paths associated with the images. The most straightforward form to represent a path is as a sequence of points on the path

$$\mathbf{M}_j = \{\boldsymbol{y}_{i,j}\}_{i=0}^{n_j}. \tag{22}$$

Here $\boldsymbol{y}_{i,j} \in \mathbb{R}^{\mathrm{d}}$ are the robot configurations, e. g. Cartesian positions or joint angles, on the $j-$th path.

To train a DNN, we need to define a loss function measuring the distance between the desired and the actual outputs and compute its gradients, which are needed to apply a back-propagation algorithm [13]. For this purpose we first convert the training data (22) to normalized AL-DMP parameters as explained in Section II-A. We obtain

$$\widetilde{\mathbf{M}}_j = \left\{\{\boldsymbol{w}_{k,j}\}_{k=1}^N,\ \boldsymbol{g}_j,\ \boldsymbol{y}_{0,j},\ \boldsymbol{y}'_{0,j}\right\}. \tag{23}$$

In general, the simplest loss function we can define to train a DNN is the mean squared error of the desired parameters and the output of the DNN:

$$\mathrm{E}_a(j) = \frac{1}{2}\left(\sum_{k=1}^{N}\|\boldsymbol{w}_k - \boldsymbol{w}_{k,j}\|^2 + \|\boldsymbol{g} - \boldsymbol{g}_j\|^2 + \|\boldsymbol{y}_0 - \boldsymbol{y}_{0,j}\|^2 + \|\boldsymbol{y}'_0 - \boldsymbol{y}'_{0,j}\|^2\right), \tag{24}$$

where $\{\{\boldsymbol{w}_k\}_{k=1}^N,\ \boldsymbol{g},\ \boldsymbol{y}_0\ \ \boldsymbol{y}'_0\}$ are the output parameters computed by the DNN and $\{\{\boldsymbol{w}_{k,j}\}_{k=1}^N, \boldsymbol{g}_j,\ \boldsymbol{y}_{0,j}\ \ \boldsymbol{y}'_{0,j}\}$ the parameters of the j-th normalized AL-DMP computed from the training dataset (23).

While loss function (24) provides for an easy implementation because its gradients are trivial to compute, it does not measure a real physical difference between the training movement and the movement calculated by the DNN, but

rather the difference between the parameters of the underlying representation. A more physically meaningful metric can be defined by directly measuring the difference between the paths $\boldsymbol{y}(x(s))$ specified by normalized AL-DMPs, which are computed by the DNN, and the training data $\boldsymbol{y}_{i,j}$ from (22). We define the following loss function

$$\mathrm{E}_p(j) = \frac{1}{2n_j} \sum_{i=1}^{n_j} \|\boldsymbol{y}(x_{i,j}) - \boldsymbol{y}_{i,j}\|^2, \tag{25}$$

where $\boldsymbol{y}(x_{i,j})$ and $x_{i,j} = x(s_{i,j})$ are obtained by integrating the normalized AL-DMP calculated by the DNN.

It is more difficult to compute the gradients of (25) with respect to the normalized AL-DMP parameters than of (24) because $\boldsymbol{y}(x_{i,j})$ are calculated by integrating differential equations (12) – (14). In the following we denote by $p_a,\ a = 1, \ldots, A$, any of the normalized AL-DMP parameters specified in Eq. (23). The partial derivatives of $\mathrm{E}_p(j)$ with respect to each normalized AL-DMP parameter p_a can be calculated as follows

$$\frac{\partial \mathrm{E}_p}{\partial p_a}(j) = \frac{1}{n_j} \sum_{i=1}^{n_j} \left(\boldsymbol{y}(x_{i,j}) - \boldsymbol{y}_{i,j}\right)^{\mathrm{T}} \frac{\partial \boldsymbol{y}}{\partial p_a}(x_{i,j}), \tag{26}$$

where $\partial \boldsymbol{y}/\partial p_a$ is the partial derivative of $\boldsymbol{y}$ with respect to the normalized AL-DMP parameter p_a.

Next we show how to compute the partial derivatives $\partial \boldsymbol{y}/\partial p_a$. Using notation $\boldsymbol{y} = [y_1, \ldots, y_d]^{\mathrm{T}}$, we observe that for any parameter p_a and any continuously differentiable path, the following holds true for partial derivatives $\partial y_l/\partial p_a$, $\partial z_l/\partial p_a$, $l = 1, \ldots, d$,

$$\begin{aligned} \frac{d}{ds}\frac{\partial}{\partial p_a} z_l &= \frac{\partial}{\partial p_a}\frac{d}{ds} z_l = \frac{\partial z'_l}{\partial p_a}, \\ \frac{d}{ds}\frac{\partial}{\partial p_a} y_l &= \frac{\partial}{\partial p_a}\frac{d}{ds} y_l = \frac{\partial y'_l}{\partial p_a}. \end{aligned}$$

Thus $\partial y_l/\partial w_{l,k}$ and $\partial z_l/\partial w_{l,k}$ can be obtained by calculating the derivatives of (12) and (13) with respect to $w_{l,k}$

$$\frac{\partial z'_l}{\partial w_{l,k}} = \alpha_z(-\beta_z \frac{\partial y_l}{\partial w_{l,k}} - \frac{\partial z_l}{\partial w_{l,k}}) + \frac{\psi_k(x)}{\sum_{n=1}^{N} \Psi_n(x)} x, \tag{27}$$

$$\frac{\partial y'_l}{\partial w_{l,k}} = \frac{\partial z_l}{\partial w_{l,k}}, \tag{28}$$

and integrating the resulting differential equation system of $\partial y_l/\partial w_{l,k}$ and $\partial z_l/\partial w_{l,k}$ with initial values

$$\frac{\partial y_l}{\partial w_{l,k}}(1) = \frac{\partial z_l}{\partial w_{l,k}}(1) = 0. \tag{29}$$

The partial derivatives with respect to the parameters $g_l,\ y_{0,l},\ y'_{0,l},\ l = 1, \ldots, d$, are obtained analogously, i. e. by calculating the partial derivatives of Eqs. (12) and (13) with respect to g_l, $y_{0,l}$ and $y'_{0,l}$. This results in

$$\frac{\partial z'_l}{\partial g_l} = \alpha_z \left(\beta_z \left(1 - \frac{\partial y_l}{\partial g_l}\right) - \frac{\partial z_l}{\partial g_l}\right), \tag{30}$$

$$\frac{\partial y'_l}{\partial g_l} = \frac{\partial z_l}{\partial g_l}, \tag{31}$$

and

$$\frac{\partial z'_l}{\partial y_{0,l}} = \alpha_z \left(-\beta_z \frac{\partial y_l}{\partial y_{0,l}} - \frac{\partial z_l}{\partial y_{0,l}}\right), \tag{32}$$

$$\frac{\partial y'_l}{\partial y_{0,l}} = \frac{\partial z_l}{\partial y_{0,l}}, \tag{33}$$

and finally

$$\frac{\partial z'_l}{\partial y'_{0,l}} = \alpha_z \left(-\beta_z \frac{\partial y_l}{\partial y'_{0,l}} - \frac{\partial z_l}{\partial y'_{0,l}}\right), \tag{34}$$

$$\frac{\partial y'_l}{\partial y'_{0,l}} = \frac{\partial z_l}{\partial y'_{0,l}}. \tag{35}$$

The values of the above partial derivatives at phases $x_{i,j}$ can be calculated by respectively integrating the equation systems (30) – (31), (32) – (33) and (34) – (35), where the initial values are set as follows

$$\frac{\partial y_l}{\partial g_l}(1) = \frac{\partial z_l}{\partial g_l}(1) = 0, \tag{36}$$

$$\frac{\partial y_l}{\partial y_{0,l}}(1) = 1,\ \frac{\partial z_l}{\partial y_{0,l}}(1) = 0, \tag{37}$$

$$\frac{\partial y_l}{\partial \tilde{y}'_{0,l}}(1) = 0,\ \frac{\partial \tilde{z}_l}{\partial \tilde{y}'_{0,l}}(1) = 1. \tag{38}$$

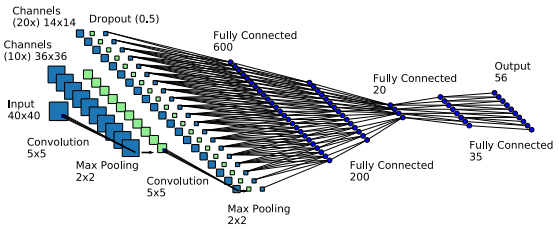
In equation (37) we took into account that y_l is initially set to $y_{0,l}$ and in equation (38) that y'_l is initially set to $y'_{0,l}$.

If we used differential equation system (5) – (7) instead of (12) – (14), we would need to calculate also the partial derivatives with respect to the parameter L. It turns out that since L affects all dimensions of the control variable $\boldsymbol{y}$, the calculation of partial derivatives with respect to L is much more complicated than for other AL-DMP parameters.

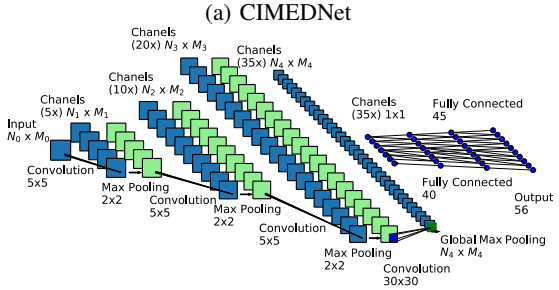
IV. Neural network architectures

In our previous work we developed two DNN architectures capable of mapping raw images to standard DMPs: the fully connected Image to Motion Encoder Decoder Network (IMEDNet) and the Convolutional Image to Motion Encoder Decoder Network (CIMEDNet) [3]. Both networks contain a bottleneck layer with fewer neurons than at the input and output of the network. The encoder part of each network maps the inputs to the bottleneck layer neurons, while the decoder part maps the bottleneck layer to the output neurons.

The architectures described above are applicable only if the input images have a fixed size. However, it is often useful to process images of different sizes because training images might come from different sources and rescaling to a fixed size results in a loss of information. In this paper, we propose a new architecture called VIMEDNet, which is based on CIMEDNet (showed in Fig. 2a) and enables the processing of variable size images. The ability to process input images of varying sizes is brought into VIMEDNet by adding a global max pooling layer instead of fully connected layers in the encoder part of the network (see Fig. 2b). The global max pooling layer chooses the maximum value from each channel and computes the bottleneck layer neurons from these values. This way we obtain a fixed number of neurons in the bottleneck layer



(a) CIMEDNet



(b) VIMEDNet

Fig. 2: CNN architectures used in our experiments. Note that the size of channels in the convolutional part of VIMEDNet network is not fixed. Thus this network can take images of any size as input.

because the number of channels is constant. We also inserted two additional convolutional and max pooling layers into the encoder part of the network and one additional fully connected layer into the decoder part (see Fig. 2b). Finally, we increased the number of neurons in the bottleneck layer from 20 to 35.

V. Execution of normalized AL-DMPs

Since robots are controlled at constant time steps, the information about time needs to be added to normalized AL-DMPs to create an executable robot motion trajectory. The normalized arc length parameter s starts at 0 and reaches 1 at the end of the path. To define a schedule to execute a normalized AL-DMP with a robot, we specify the desired duration of motion T and compute a fifth order polynomial $s(t)$ with boundary conditions $s(0) = \dot{s}(0) = \ddot{s}(0) = \dot{s}(T) = \ddot{s}(T) = 0,\ s(T) = 1$. Note that $\dot{s}(t) > 0$ for all $0 < t < T$.

To sample the path at the constant time step Δt, $t_{i+1} = t_i + \Delta t$, $t_0 = 0$, we first compute the integration step

$$\Delta s_i = s(t_i + \Delta t) - s(t_i). \tag{39}$$

We can then integrate the normalized AL-DMP with a variable integration step Δs_i to sample the corresponding time parametrized trajectory at constant time steps as required by robot controllers. The desired positions $\boldsymbol{y}$ are obtained directly by integrating (12) – (14), while the velocities $\dot{\boldsymbol{y}}$ and accelerations $\ddot{\boldsymbol{y}}$ (if needed) can be computed as follows

$$\dot{\boldsymbol{y}} = \frac{d}{dt}\boldsymbol{y}(s(t)) = \boldsymbol{y}'\dot{s}, \tag{40}$$

$$\ddot{\boldsymbol{y}} = \frac{d}{dt}(\boldsymbol{y}'(s(t))\,\dot{s}(t)) = \boldsymbol{y}''\dot{s}^2 + \boldsymbol{y}'\ddot{s}, \tag{41}$$

Note that the choice of schedule $s(t)$ does not change the spatial course of motion. Any continuously increasing time parametrization of s results in the same spatial path, but executed at different speeds. Compared to standard DMPs [4], the normalized AL-DMPs as defined by Eqs. (12) – (14) do not enable temporal and spatial scaling of the resulting motion. If these properties of DMPs are needed, we can sample the normalized AL-DMP with the given schedule $s(t)$ using the above procedure and re-encode the resulting motion using the standard DMP representation.

VI. Experimental evaluation

We tested the developed DNNs and criterion functions on the problem of reproducing handwritten digits. We first considered the effectiveness of different movement representations and loss functions by comparing the performance of the proposed approach when using standard DMPs and normalized AL-DMPs. In the second set of experiments, we evaluated the performance of CIMEDNet and VIMEDNet on input images of different sizes. Finally, we tested the performance of the approach when real images taken by a humanoid robot are used as input. In all cases, we trained the DNNs with the same training parameters as in [3]. We applied dynamic time warping (DTW) [14] to accurately measure the quality of handwriting reproduction.

A. Datasets

We developed several datasets for testing purposes. The first dataset called *synthetic MNIST (s-MNIST)* was developed in our previous work [3] and takes inspiration from the well-known MNIST dataset [15]. To generate s-MNIST, the handwriting movements and the associated images are computed synthetically from geometric primitives (lines and ellipses) with variable parameters that are varied randomly.

Since sharp corners result in discontinuous velocities, AL-DMPs cannot represent such paths. With standard DMPs and real physical movements, sharp corners are handled by reducing the speed of movement, but AL-DMPs cannot do this because the derivatives with respect to the arc length parameter always have unit norm. In our experiments we therefore recreated only the digits without sharp corners: 0, 6, 8 and 9. See Section VII for the discussion of how to deal with other digits using AL-DMP representation and why AL-DMPs are anyway advantageous compared to DMPs. For each digit, we generated 5000 pairs of handwriting movements (defined as a sequence of 2-D points on the trajectory) and the corresponding 40×40 pixel binary images of digits. The resulting dataset altogether contains 20000 pairs of images and the corresponding paths, as specified in (21). Paths are sampled at equidistant points, with the sampling step Δs equal to approximately 0.2 pixels, where the exact sampling step is computed differently for each path so that the path endpoints are always included. The constant sampling step is fine because our data are synthetic and do not contain noise.

To simulate variability in the temporal execution of motion typical for humans, we created a second dataset, which consists of the same spatial trajectories (paths) as s-MNIST, but with nonlinearly scaled temporal course of motion. At the beginning of Section V, we defined the normalized arc length as a function of time, which is given by a fifth order polynomial $s(t)$. This parametrization was used to generate

s-MNIST. For this new dataset, we scaled $s(t)$ by a quadratic function $g(s(t)) = as(t)^2 + (1-a)s(t)$ with randomly selected parameter a, where $-1 \leq a \leq 1$. The resulting parametrization is still strictly increasing but is nonlinearly transformed in time. While such a transformation does not change the weights $\boldsymbol{w}$ of normalized AL-DMPs, it does change the weights of standard DMPs. We call this dataset *temporally scaled s-MNIST (s-MNIST-TS)*. The important difference between s-MNIST and s-MNIST-TS is that in the former, the selection of duration of motion does not affect the weights of DMPs, whereas in the latter it does.

The third dataset we developed is also based on s-MNIST. It was generated by pasting 40×40 s-MNIST images onto images with cluttered background of size 80×80 pixels. This way we ensured that in every image comprising the dataset, there is a portion of image that is not covered by a 40x40 image of a digit. The LSUN-dataset [16] was taken as a source of background images. From this dataset, we took 300 images belonging to the scene "Classroom" and resized them to the resolution 80×80. The resulting background images were filtered with a Gaussian filter. The handwriting path for each example had two versions, one the same as in s-MNIST and another as in s-MNIST-TS. We call the resulting datasets *s-MNIST / s-MNIST-TS with background*.

To train DNNs suitable for processing real images captured by a robot, we generated the fourth dataset based on s-MNIST. In this dataset, the width of curves forming the images of digits varied and the simulated images of digits were computed as grayscale instead of binary images. We call this dataset *grayscale s-MNIST with background*. In all datasets, 5000 pairs of handwriting movements and digit images were generated per digit, thus altogether they contain 20000 image-handwriting movements pairs.

B. Reproduction of handwritten digits

First we evaluated the effectiveness of different representations and network architectures for the reproduction of handwritten digits. We applied 4 different loss functions to train the DNNs: (24) and (25) for networks with normalized AL-DMP parameters at the output and the equivalent loss functions for networks that compute the DMP parameters. The details of DMP-based loss functions are described in [3], but their forms are similar to (24) and (25).

The results are presented in Tab. I. They show the average distance of handwriting paths generated by DMPs and normalized AL-DMPs computed by the respective DNNs from the paths in the test dataset. 70% of all data were used for training, 15% for validation, and 15% for testing. DTW was applied to compute the distances between trajectories in pixels. Although CIMEDNet contains significantly less parameters than IMEDNet, the performance of both network architectures was similar when using the same representation and loss function. Note, however, that CIMEDNet can be trained significantly faster than IMEDNet.

The best results were obtained when using normalized AL-DMP representation with loss function (25) and the second best results with DMP representation and the DMP version of loss function (25). The reason for this is that unlike loss function (24), which measures the distance between the normalized AL-DMP parameters that have no physical meaning, loss function (25) measures the real spatial distance between the digits. The performance of normalized AL-DMPs is better because the timing of motion does not affect the result. Consequently, in the normalized AL-DMP representation, the points are evenly distributed along the path, whereas in the DMP representation, the trajectory points are more distant and less numerous in parts with high velocity. Thus the precision of approximation is lower with DMP-based DNNs in high velocity parts of trajectories.

TABLE I: Reconstruction statistics for test data with DMP and normalized AL-DMP parameters at the DNN output when using different loss functions. The results are in pixels.

IMEDNet	
s-MNIST & DMP parameters loss function from [3]	0.116 ± 0.028
s-MNIST & DMP trajectory loss function from [3]	0.088 ± 0.016
Temp. scal. s-MNIST & DMP traj. loss fun. from [3]	2.279 ± 0.486
s-MNIST & AL-DMP parameters loss function (24)	0.264 ± 0.074
s-MNIST & AL-DMP path loss function (25)	0.059 ± 0.014
Temp. scal. s-MNIST & AL-DMP path loss fun. (25)	0.059 ± 0.014
CIMEDNet	
s-MNIST & DMP parameters loss function from [3]	0.116 ± 0.029
s-MNIST & DMP trajectory loss function from [3]	0.091 ± 0.020
Temp. scal. s-MNIST & DMP traj. loss fun. from [3]	2.220 ± 0.516
s-MNIST & AL-DMP parameters loss function (24)	0.182 ± 0.087
s-MNIST & AL-DMP path loss function (25)	0.066 ± 0.017
Temp. scal. s-MNIST & AL-DMP path loss fun. (25)	0.066 ± 0.017

While the performance of estimation with loss function (24) and its DMP equivalent from [3] was worse for both network architectures, the DMP representation outperformed the normalized AL-DMP representation in this case. We believe that this reversal of performance is due to the additional parameters of normalized AL-DMPs, i. e. the starting velocities, which have different scale than position parameters and must be normalized. Since there are more parameters in normalized AL-DMPs, such scaling problems negatively affect the performance of networks with AL-DMP outputs compared to the networks with DMP outputs.

The above results already show the advantages of AL-DMPs in combination with path loss function (25) over standard DMPs. But the most important advantage of AL-DMPs is their independence from the temporal schedules. When the proposed DNNs are trained using data with inconsistent temporal schedules, e. g. temporally scaled s-MNIST data, DMP-based DNNs are no longer able to learn handwritten digits precisely, as evident from Table I. On the other-hand, the results of normalized AL-DMPs are not affected by the temporal inconsistencies at all. In this case, the results are by an order of magnitude better with normalized AL-DMP- than with DMP-based networks.

C. Processing input images of different sizes

In the next experiment we tested the performance of both architectures when processing images of variable size, with the networks trained using either the original or temporally

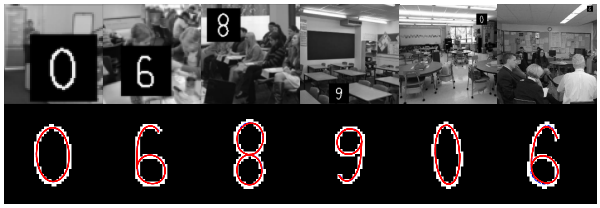


Fig. 3: Visualization of writing trajectories generated by VIMEDNet with input images of different sizes. 40×40 images of digits were pasted onto $60 \times 60, 80 \times 80, 120 \times 120, 200 \times 200, 360 \times 360$, and 680×680 pixels background images. The DNN was trained with 80×80 pixel images.

scaled handwriting trajectories. The s-MNIST and s-MNIST-TS datasets with background were used for training in these experiments (see Section VI-A).

We trained VIMEDNet on 80×80 pixel images, but we used images of different sizes for testing. The background images in the training set were different from the ones in the testing set. For comparison, we also trained and tested the CIMEDNet architecture. Since CIMEDNet cannot deal with variable size images, we had to train a different DNN for each image size.

As evident from Fig. 3, VIMEDNet is able to reproduce good quality writing movements even for inputs with different image sizes although it was trained only on 80×80 pixel images. Results in Tab. II again confirm that the DMP representation cannot handle temporally scaled handwriting trajectories, but the AL-DMP representation has no problems with such trajectories. The performance of VIMEDNet was in all cases better than the performance of CIMEDNet.

Finally, the results in Tab. II show that the performance of CIMEDNet is good for small inputs. However, as the input image size increases, the performance drops drastically because the number of CIMEDNet parameters depends quadratically on the size of the input. This makes training more difficult and computationally expensive for larger inputs. With the available resources, it was not possible to train CIMEDNet with input images larger than 120×120 pixels.

D. Experiment with real robot and images

VIMEDNet can reproduce digits from camera images of any size. However, the size of subimages containing the digits must be similar as in the training data. On the other hand,

TABLE II: Reconstruction statistics for test data using input images of different sizes. The results are in pixels.

		Temp. scal. DMP trj.	**No temp. scal. DMP trj.**	**Temp scal. AL-DMP path**
CIMEDNet	60×60	2.223 ± 0.472	0.477 ± 0.198	0.247 ± 0.093
	80×80	4.613 ± 1.761	0.709 ± 0.261	0.419 ± 0.164
	120×120	5.271 ± 1.018	5.640 ± 1.042	2.919 ± 2.239
VIMEDNet	60×60	2.398 ± 0.554	0.149 ± 0.077	0.146 ± 0.049
	80×80	2.347 ± 0.496	0.136 ± 0.074	0.138 ± 0.043
	120×120	2.326 ± 0.483	0.134 ± 0.061	0.135 ± 0.041
	200×200	2.325 ± 0.481	0.139 ± 0.089	0.136 ± 0.043
	360×360	2.324 ± 0.475	0.166 ± 0.162	0.138 ± 0.050
	680×680	2.317 ± 0.474	0.199 ± 0.215	0.139 ± 0.049

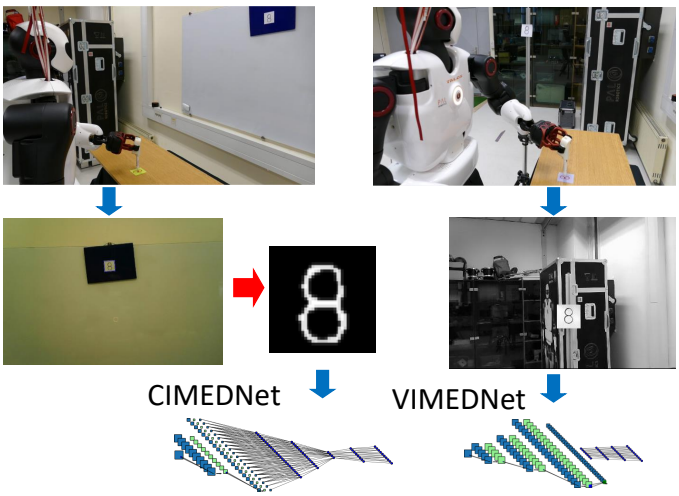


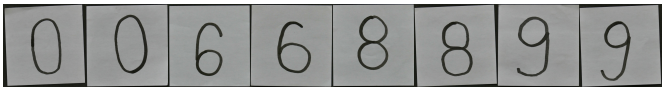
Fig. 4: Comparison of the experimental setup for testing CIMEDNet (left) and VIMEDNet (right). For CIMEDNet, the digit subimage is first extracted from the camera image using computer vision algorithms (red arrow) and the resized digit image is used as input. For VIMEDNet, the acquired camera image is used directly as input to the network.

with CIMEDNet both the size of input images and the size of subimages containing the digits must be the same or similar as in the training data. This is a severe limitation of CIMEDNet because one cannot simply rescale camera images to a smaller size. By doing so, the digit size could become too small and the digit no longer recognizable.

To overcome this issue, we generated inputs for CIMEDNet by processing real camera images with standard computer vision algorithms in order to detect the area containing the digit in the acquired image. The extracted subimage was then resized to the image size for which CIMEDNet had been trained. For these tests, we trained CIMEDNet with images of fixed size (40×40 pixels) using s-MNIST dataset.

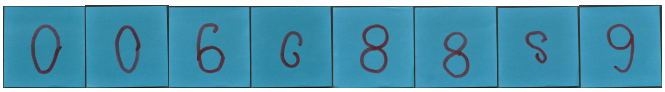
In the experiment with a real robot, we evaluated end-to-end reproduction of writing trajectories from real images. We pasted a piece of paper with a handwritten digit onto the rod or whiteboard in front of the robot (see the upper row of Fig. 4). The real images were taken by an on-board camera of a humanoid robot TALOS. VIMEDNet was trained on grayscale s-MNIST with background dataset, where the size of subimages containing the digits is 40×40. We therefore made sure that the size of subimage containing the sheet of paper with a digit was approximately 40×40 pixels. To compute the inputs for CIMEDNet, we used images of digits pasted on the whiteboard where segmentation is easier than with more complex backgrounds.

Robot-written digits are shown in Fig. 5, where both types of DNNs computed AL-DMP outputs. The comparison of human- and robot-written digits makes it clear that VIMEDNet was able to reproduce the handwritten digits, but in this case the reproduction quality was somewhat worse than that of CIMEDNet. This is because CIMEDNet received segmented digits and thus had better data to work with. As shown in Section VI-C, VIMEDNet would outperform CIMEDNet if given the same data. Nevertheless, the results prove that it is possible



(a) Handwritten digits shown to the robot.

(b) Digits written by TALOS using the output of CIMEDNet, where input images were segmented to contain only the image of a digit.



(c) Digits written by TALOS using the output of VIMEDNet, where unprocessed camera images were fed to the network as input.

Fig. 5: Reproduction of handwritten digits with TALOS

to train DNNs on synthetic data with input images smaller than the processed real images and that VIMEDNet can reproduce digits without applying standard computer vision algorithms to prepare the input.

VII. Conclusions and Future Work

The most important result of this paper is the extension of our approach for training of DNNs that output DMP parameters [3], to training of DNNs that output normalized AL-DMP parameters. For this purpose, we derived the gradient of the loss function for normalized AL-DMPs, which measures the real physical distance between training paths and the paths generated by normalized AL-DMPs. This gradient is needed to enable the application of the backpropagation algorithm for DNN training. We confirmed experimentally that the performance of DNNs that output normalized AL-DMP parameters can be significantly improved by the proposed loss function. We showed that DNNs based on normalized AL-DMPs outperform the DNNs based on standard DMPs by an order of magnitude in cases where the training data contains trajectories with varying speed but the same path.

Compared to DMPs, AL-DMPs and the normalized AL-DMPs are limited to paths with no sharp corners. This is because a real robot must stop if it is to change its direction of motion abruptly. This is handled in standard DMPs by reducing the speed of motion, but speed is separate in AL-DMPs. This limitation can be overcome by segmenting paths into parts without sharp corners and describing each segment of the path with a separate AL-DMP. On the other hand, DMPs offer no solution to deal with trajectories of varying speed, which often occur in handwriting. Thus the AL-DMP representation is still advantageous.

The proposed architectures offer no direct way to generate a different number of AL-DMPs on the DNN output. A possible solution is to first train a classification DNN that identifies the observed digit and then apply DNNs trained for each specific digit. For example, a DNN trained for digit five could output three AL-DMPs as there are two sharp corners in digit five. Another possibility is to use an architecture similar to the one proposed by He et al. [11], where a variable number of parts from the input feature map can be used for prediction. Such an architecture would enable us to deal with multiple digits in the input image or generate a variable number of handwriting movements to avoid sharp corners.

The second contribution of the paper is the development of VIMEDNet architecture, which takes images of different sizes as input and outputs either DMP or normalized AL-DMP parameters. The proposed architecture can handle images with a significant portion of background pixels that do not belong to the observed digit. VIMEDNet can be trained on small images and then applied to larger inputs.

Besides experiments with synthetic data, we also performed real robot writing tests, where we showed that VIMEDNet can reproduce the observed digits. In these experiments we assumed that the orientation and size of digits in the input images were constant. This limitation can be resolved by combining VIMEDNet with the approach of Ridge et al. [17], which can handle variations in digit size and orientation but cannot deal with varying size input images.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] N. Krüger, C. Geib, J. Piater, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrčen, A. Agostini, and R. Dillmann, "Object-Action Complexes: Grounded abstractions of sensory-motor processes," *Robotics and Autonomous Systems*, vol. 59, no. 10, pp. 740–757, 2011.

[3] R. Pahič, B. Ridge, A. Gams, J. Morimoto, and A. Ude, "Training of deep neural networks for the generation of dynamic movement primitives," *Neural Networks*, vol. 127, pp. 121–131, 2020.

[4] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[5] Y. Zhou, J. Gao, and T. Asfour, "Movement primitive learning and generalization: Using mixture density networks," *IEEE Robotics & Automation Magazine*, pp. 2–12, 2020.

[6] A. Ude, R. Vuga, B. Nemec, and J. Morimoto, "Trajectory representation by nonlinear scaling of dynamic movement primitives," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, 2016, pp. 4728–4735.

[7] T. Gašpar, B. Nemec, J. Morimoto, and A. Ude, "Skill learning and action recognition by arc-length dynamic movement primitives," *Robotics and Autonomous Systems*, vol. 100, pp. 225–235, 2018.

[8] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 297–330, 2020.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *arXiv e-prints*, p. arXiv:1406.4729, 2014.

[10] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv e-prints*, p. arXiv:1312.4400, 2013.

[11] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *IEEE Int. Conf. on Computer Vision (ICCCV)*, 2017, pp. 2980–2988.

[12] W. Suleiman, "On time parameterization of a robot path," *IFAC Papers OnLine*, vol. 48, no. 3, pp. 52–57, 2015.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[14] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[15] Y. LeCun, C. Cortes, and C. Burges, "The MNIST database of handwritten digits," http://yann.lecun.com/exdb/mnist/.

[16] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, "LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.

[17] B. Ridge, R. Pahič, A. Ude, and J. Morimoto, "Learning to write anywhere with spatial transformer image-to-motion encoder-decoder networks," in *IEEE International Conference on Robotics and Automation (ICRA)*, Montreal, Canada, 2019, pp. 4797–4803.