# Learning to Write Anywhere with Spatial Transformer Image-to-Motion Encoder-Decoder Networks

Barry Ridge[1,2], Rok Pahič[2], Aleš Ude[2], and Jun Morimoto[1]

***Abstract*— Learning to recognize and reproduce handwritten characters is already a challenging task both for humans and robots alike, but learning to do the same thing for characters that can be transformed arbitrarily in space, as humans do when writing on a blackboard for instance, significantly ups the ante from a robot vision and control perspective. In previous work we proposed various different forms of encoder-decoder networks that were capable of mapping raw images of digits to dynamic movement primitives (DMPs) such that a robot could learn to translate the digit images into motion trajectories in order to reproduce them in written form. However, even with the addition of convolutional layers in the image encoder, the extent to which these networks are spatially invariant or equivariant is rather limited. In this paper, we propose a new architecture that incorporates both an image-to-motion encoder-decoder and a spatial transformer in a fully differentiable overall network that learns to rectify affine transformed digits in input images into canonical forms, before converting them into DMPs with accompanying motion trajectories that are finally transformed back to match up with the original digit drawings such that a robot can write them in their original forms. We present experiments with various challenging datasets that demonstrate the superiority of the new architecture compared to our previous work and demonstrate its use with a humanoid robot in a real writing task.**

## I. INTRODUCTION

In order to enable a robot to learn how to read and write effectively, or indeed to learn how to tackle many other tasks that involve tightly coupling perception and action skills, an effective strategy is to break the overall problem down into individually soluble components before attempting to merge the individual solutions into a fully-functioning holistic approach. At the heart of the writing problem, for example, is the necessity to be able to translate between visual representations of characters perceived from raw images and the action representations needed to control the motion trajectories required for drawing them, and while this core problem can be effectively tackled individually, other important problems lurk both within it and at its periphery. In previous work [1] we directly tackled this core problem by proposing an image-to-motion encoder-decoder neural network architecture (IMEDNet) that was capable of converting raw images of digits into the dynamic movement primitives (DMPs) developed by Ijspeert et al. [2] for writing trajectory motion representation. However, given the fully-connected nature of that original network, the visual perception component was a relatively weak point, thus in the work presented here we improved upon this by including pre-trained convolutional neural network (CNN) layers in the image encoder (CIMEDNet) making the network more robust to image variation and noise.

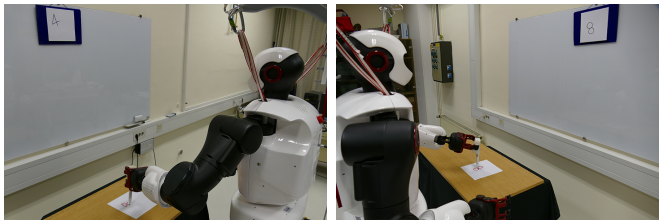

Fig. 1: Writing digits with the Talos humanoid robot using a spatial transformer image-to-motion encoder-decoder network (STIMEDNet) prediction. The movements are generated using DMP trajectories predicted by the network from the images shown to the robot and the robot can write the digits in matching poses.

While the original IMEDNet network was useful for converting images of digits into motion trajectories, certain difficulties became apparent when considering real-world scenarios in which, for example, a robot is shown a digit on a piece of paper held in front of its camera or written in free-form on a board, and must generate a corresponding motion. The networks are trained on images of a certain size with the objects generally presented in canonical poses with relatively low degrees of variation with respect to position, scale and orientation. When applying a trained network to localize such objects within a larger image, the standard technique is to employ an appropriately sized sliding window where the network is applied at multiple locations at regular intervals. While this is effective at dealing with translations, even when CNN layers are employed, as was the case with CIMEDNet, there are limitations when it comes dealing with rotation and scaling. The max-pooling layers in a CNN can provide a certain amount of invariance when it comes to such transformations, but due to their relatively small spatial support, even with very deep networks this tends not to extend to the types of large affine transformations under consideration in our chosen problem domain [3], [4].

There have been ongoing efforts to develop different forms of deep neural networks that break away from the restrictions imposed by the max-pooling layers of CNNs, capsule networks [5] being a particularly striking example, where equivariance, that is proportional transformation variation between inputs and outputs, is directly considered. Their de-

[1]ATR Computational Neuroscience Laboratories, Kyoto, Japan barry.ridge@atr.jp, xmorimo@atr.jp

[2] Dept. of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Ljubljana, Slovenia rok.pahic@ijs.si, ales.ude@ijs.si

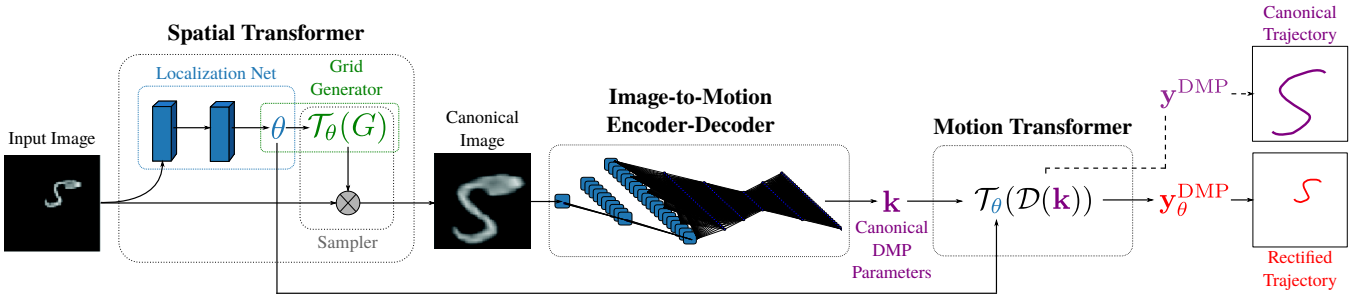


Fig. 2: The proposed STIMEDNet architecture. Input images are fed to a spatial transformer network (STN) which rectifies attended objects into canonical form before being passed to a pre-trained convolutional image-to-motion encoder-decoder network (CIMEDNet) producing DMP parameters $\mathbf{k}$ that are integrated via $\mathcal{D}$ (Euler's method) into motion trajectories $\mathbf{y}^{\mathrm{DMP}}$ and transformed via $\mathcal{T}_\theta$ to final output form $\mathbf{y}_\theta^{\mathrm{DMP}}$ by a motion transformer (MTN). The same $\theta$ transform parameters learned by the STN to transform the images at the bottom of the network are passed to the MTN via skip connection where they are reused to transform the trajectories at the top of the network. The solid arrows mark differentiable paths through the network, the dashed arrow marks auxiliary output.

sign, however, is fundamentally different to that of CNNs and their development is comparatively still in its infancy. The recently introduced *spatial transformer networks (STNs)*, on the other hand, provide a means of approaching the problem of invariance and equivariance to affine transformations, but can be relatively easily incorporated into existing or novel CNN-based architectures.

Thus, building on our own recent work and on these other recent advances, in this paper, we propose a novel architecture that intertwines three main components in order to effectively approach the proposed robot writing problem. The first of these components is an STN that reads input images containing the affine-distorted digits and learns how to transform them into images in which the digits are in canonical poses. At the core of the architecture is the second component, the IMEDNet network, which converts the canonical digit images into DMP parameters. The third and final component is a motion transformer that first integrates the DMPs into motion trajectories and then uses the same transformation parameters passed forward from the STN to transform the predicted canonical trajectories to correspond to the canonical poses of the digits as originally presented in the input images. The entire network is fully differentiable and can be trained end-to-end.

*1) Related Work:* With regard to handwriting synthesis specifically, Graves et al. [6] introduced a deep multi-layered recurrent neural network (RNN) that utilized long short-term memory (LSTM) units [7] in order to generate synthetic handwriting in point-by-point sequences. The Deep Recurrent Attentive Writer (DRAW) network [8] combined a spatial attention mechanism mimicking foveation with a sequential variational auto-encoding framework that allowed for the iterative construction of images from both the MNIST and SVNH (Google StreetView House Numbers) datasets. Ha et al.[9], proposed a sequence-to-sequence variational autoencoder where the encoder is an RNN that converts input images containing hand-drawn sketchs to vector sequences such that the network can generate its own sketches. This paper was also behind the contribution of Google's Quick-Draw dataset, a large-scale database of hand-drawn sketches that was used to train the network.

Prior to the adoption of deep learning methods for such tasks, an effort was made by Ali [10] at modeling individual brush strokes of calligraphic characters using Gaussian Mixture Models, combining the brush strokes using Gaussian Mixture Regression and reproducing brush stroke trajectories using DMPs. The reproduced stroke trajectories were iteratively refined using reinforcement learning for learning examples in the database, but each reproduction started from scratch with no generalization between different examples or to new unseen cases. Usually when CNNs are used for supervised learning of perception-action couplings, they are used in combination with another neural network in two separately trainable parts. In [11], Yang et al. first used a deep convolutional autoencoder for finding camera image features and then in combination with recorded robot angles, formed sequences for the learning task dynamics with a time delay neural network. Pervez et al. [12] used a pre-trained CNN for finding task parameters from input images, while using a another fully-connected neural network to learn to generate forcing terms from the clock signal and task parameters, before combining both networks in an end-to-end training scheme. Both of these two examples produce the next step from the image of the current step while working in online loop, whereas our method, by contrast, uses just single images for generating entire trajectories.

## II. SPATIAL TRANSFORMER IMAGE-TO-MOTION ENCODER-DECODER NETWORKS

The newly proposed spatial transformer image-to-motion encoder-decoder network (STIMEDNet) architecture is illustrated in Fig. 2. In order to describe the proposed architecture in more detail, we begin by discussing the nature of the data that it must process, the structure of which is the same as in [1]. The input and output data pairs take the following form:

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

where $M$ is the number of input and output training pairs, i. e. input images $\mathbf{C}_j \in \mathbb{R}^{H \times W}$ of width $W$ and height

$H$, and output trajectories $\mathbf{M}_j$ associated with each image, where

$$\mathbf{M}_j = \{\boldsymbol{y}_{i,j}, t_{i,j}\}_{i=1}^{T_j}. \quad (2)$$

Here $\boldsymbol{y}_{i,j} \in \mathbb{R}^{\mathrm{d}}$ are the vectors describing the movement's degrees of freedom, e. g. Cartesian positions or joint angles, $t_{i,j} \in \mathbb{R}$ the measurement times for the $j$-th movement, and $d$ is the number of degrees of freedom. However, it should be noted that in this paper, we convert the movements $\mathbf{M}_j$ to DMPs and construct all of the datasets used to train the network models as follows:

$$\mathbf{D}' = \{\mathbf{C}_j, \mathbf{k}_j\}_{j=1}^{M}, \quad (3)$$

where $\mathbf{k}_j$ are the DMP parameters calculated for each movement $\mathbf{M}_j$ and are represented as follows:

$$\mathbf{k}_j = \left\{\{\boldsymbol{w}_k\}_{k=1}^{N},\ \tau,\ \boldsymbol{g},\ \boldsymbol{y}_0\right\}. \quad (4)$$

The construction of DMPs and the nature of the parameters $\{\boldsymbol{w}_k\}_{k=1}^{N}$, $\tau$, $\boldsymbol{g}$ and $\boldsymbol{y}_0$ are briefly explained further below in the following section.

### A. Motion Representation with DMPs

Letting any given motion trajectory be denoted as $\boldsymbol{y}(t) \in \mathbb{R}^d$, a DMP specifying this motion is defined by a system of differential equations

$$\tau \dot{\boldsymbol{z}} = \alpha_z(\beta_z(\boldsymbol{g} - \boldsymbol{y}) - \boldsymbol{z}) + \mathrm{diag}(\boldsymbol{g} - \boldsymbol{y}_0)\mathbf{F}(x), \quad (5)$$

$$\tau \dot{\boldsymbol{y}} = \boldsymbol{z}. \quad (6)$$

This equation system contains several parameters including the initial position $\boldsymbol{y}_0 \in \mathbb{R}^d$ on the trajectory and the final position $\boldsymbol{g} \in \mathbb{R}^d$. The diagonal matrix $\mathrm{diag}(\boldsymbol{g} - \boldsymbol{y}_0) \in \mathbb{R}^{d \times d}$ contains the coefficients of vector $\boldsymbol{g} - \boldsymbol{y}_0$ on its diagonal. The auxilliary parameter $\boldsymbol{z} \in \mathbb{R}^d$ is a scaled velocity of motion, while $x \in \mathbb{R}$ is the phase defined by the following equation

$$\tau \dot{x} = -\alpha_x x. \quad (7)$$

The phase is used instead of time to avoid explicit time dependency. It is fully defined by setting its initial value to $x(0) = 1$. Eq. (5) also contains the nonlinear forcing term $\mathbf{F}(x)$, which is usually defined as a linear combination of RBFs (radial basis functions)

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

$$\Psi_k(x) = \exp\left(-h_k\left(x - c_k\right)^2\right). \quad (9)$$

Here $c_k$ are the centers of RBFs distributed along the phase of the trajectory, and $h_k$ their widths. The role of $\mathbf{F}$ is to adapt the dynamics of the linear part of Eq. (5) – (6) to any desired trajectory $\boldsymbol{y}$. This way the dynamic system can reproduce any smooth motion between the starting and final robot configuration $\boldsymbol{y}_0$ and $\boldsymbol{g}$, respectively.

The linear part of equation system (5) – (6) is critically damped if the fixed parameters $\tau, \alpha_x, \alpha_z, \beta_z \in \mathbb{R}$ fullfil $\tau, \alpha_x > 0$ and $\alpha_z = 4\beta_z > 0$. In this case $\boldsymbol{y}, \boldsymbol{z}$ monotonically converge to a unique attractor point at $\boldsymbol{y} = \boldsymbol{g}$, $\boldsymbol{z} = 0$.

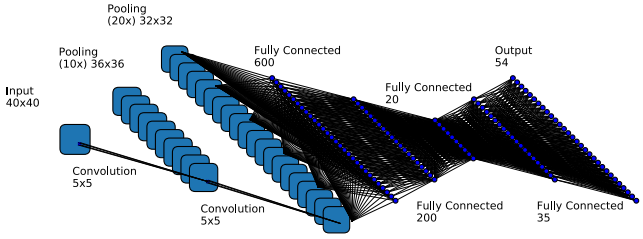


Fig. 3: The image-to-motion encoder-decoder network (CIMEDNet) at the core of the overall architecture shown in Fig. 2 with an encoder consisting of two convolutional layers followed by two fully-connected layers, and a decoder consisting of three fully-connected layers.

Standard regression techniques are usually applied to compute the free parameters $\boldsymbol{w}_k \in \mathbb{R}^d$ [13]. Constants $\alpha_z$, $\beta_z$, and $\alpha_x$ are set to fixed values for all movements. All other parameters of differential equation system (5) – (7), i. e. $\boldsymbol{w}_k$, $\tau$, $\boldsymbol{g}$ and $\boldsymbol{y}_0$ need to be computed by the neural network.

### B. Network Architecture

*1) Spatial Transformer:* The spatial transformer at the base of the architecture in Fig. 2 is an implementation of the spatial transformer network originally proposed in [14] that takes as input an image containing an affine-transformed digit, or potentially some other type of character or object, passes it through a localization network in order to learn the transformation parameters $\theta$, and transforms it into an output image with the attended digit in canonical form. This is achieved by using a sampling grid $G$ of normalized coordinates that is placed over the output image and is projected back to the input image via the transform $\mathcal{T}_\theta(G)$ before a bilinear sampler is used to sample points from the grid in the input image in order to generate the pixels in the output image. If we can assume that the motion trajectories $\mathbf{M}_j$ in our training data from Equation (1) can be matched to equivariant transformations of the DMP-integrated trajectories predicted from the canonical images, which is a reasonable assumption in the problem domain under consideration in this paper, then we can reuse the $\theta$ parameters to transform the trajectories from their canonical form to their output form later in the network in the motion transformer. Thus we pass these $\theta$ parameters directly to the motion transformer via skip connection, taking advantage of the fact that gradients can be propagated from the top of the network to the bottom via this pathway in order to potentially learn the requisite transformations faster.

*2) Image-to-Motion Encoder-Decoder:* The image-to-motion encoder-decoder network that lies at the heart of the architecture presented in Fig. 2 is a convolutional version of the fully-connected IMEDNet from [1]. It uses both convolutional and fully-connected layers in the image encoder and fully-connected layers in the DMP decoder. It is illustrated in more detail in Fig. 3. The convolutional layers are pre-trained as part of a basic CNN classifier that was trained on the original MNIST dataset [15] The input is a $40 \times 40 \times 1$ grayscale pixel image, followed the encoder consisting of a

convolutional layer with $5 \times 5$ kernel size and $10$ feature maps, a convolutional layer with $5 \times 5$ kernel size and $20$ feature maps, and two fully-connected layers with sizes of $600$ neurons and $200$ neurons respectively. Following these fully-connected layers, at the bottleneck of the network that forms the latent space representation, a decoder is formed with more fully-connected layers that gradually expand the number of units in each layer until the final output layer, which has a size set to $54$ units in order to match the DMP parameters specified in Equation (4). The layers of the decoder are illustrated on the right side of 3 starting with the bottleneck of size $20$, followed by a layer of size $35$ and finishing with the output layer. This is the same decoder structure as used [1] and we retain it here as-is, having found it to be effective throughout our experiments for this particular use case. It should be noted, however, that there are no particular restrictions on the nature of the network that is used here, and it is easy to imagine a more advanced pre-trained CNN being used for the image encoder, for example, if the application demanded it.

### C. Cost Function and Gradients

The cost function used to evaluate the output of the network and for applying backpropagation [16] in order to learn the network weights, is a slightly modified form of Equation (10) from [1], which directly measures the difference between the final transformed trajectories $\boldsymbol{y}_\theta^{\mathrm{DMP}}$ output by the motion transformer module and the $\boldsymbol{y}_{i,j}$ training data from Eq. (2), and is defined as follows:

$$\mathrm{E}_t(j) = \frac{1}{2T_j} \sum_{i=1}^{T_j} \|\boldsymbol{y}_\theta^{\mathrm{DMP}}(x_{i,j}) - \boldsymbol{y}_{i,j}\|^2. \qquad (10)$$

This differs from the cost function of Equation (9) from [1] which measured the distance between the DMP parameters $\mathbf{k}$ predicted by the encoder-decoder network and the $\mathbf{k}_j$ DMP parameters from the training data in Equation (3). We must use the former cost function here because we require that the trajectories be produced via DMP integration within the network such that they can be transformed at the top of the network, which is trained end-to-end. The calculation of the gradients of the cost function in Equation (10) is not as straightforward as for the cost function that uses the DMP parameter distances, but since the $\theta$-transform of the trajectory does not cause the calculation to differ from that of Equation (10) in [1], we refer the interested reader to that paper for further details.

## III. EXPERIMENTS

To evaluate the proposed model, we performed experiments involving synthetically generated datasets and a proof-of-concept implementation using the full-scale Talos humanoid robot. In our experiments on the synthetic datasets, we trained the IMEDNet network, the CIMEDNet network, and the proposed newly proposed STIMEDNet architecture on various digit image and motion trajectory datasets. The IMEDNet architecture was the same as described in [1] with fully-connected hidden layer sizes of 1500, 1300, 1000, 600, 200, 20, and 35 neurons, respectively. The CIMEDNet architectures was as described in Section II-B and as illustrated for in in Fig. 3. We used PyTorch [17] in order to implement all networks[1] and trained our models on NIVIDIA GTX 1080 GPUs. When pre-training the MNIST CNN classifier described in Section II-B.2, we used a stochastic gradient descent optimizer, a negative log-likelihood loss, a batch size of 64, a learning rate of 0.01 with momentum 0.5, and trained for 10 epochs. This achieved a $98\%$ accuracy which we deemed sufficient for our purposes in extracting the trained convolutional layers for use in the encoder-decoder networks.

When training the IMEDNet, CIMEDNet and STIMEDNet networks, we used the Adam optimizer [18] with a learning rate of $0.0005$, with the epsilon parameter for numerical stability set to $0.001$ and with weight decay, which provides L2 regularization equivalence, to $0.000025$. In order to avoid learning plateaus, the optimizer parameters were periodically reset to initial values every 500 epochs. A mean squared error loss was used to evaluate the cost function in Equation (10) and the batch size was set to 140 for weight updates. As a stopping criterion, if the best validation loss was unchanged after 60 epochs, training was halted. The above training procedure was used for all three of the IMEDNet, CIMEDNet and STIMEDNet architectures. The pre-trained MNIST CNN model was used to initialize the weights of the convolutional layers in the CIMEDNet model, as described previously, and a pre-trained CIMEDNet model was used to initialize the weights of the CIMEDNet model contained within the STIMEDNet architecture. All three models were fine-tuned with end-to-end training via backpropagation on the training sets from the datasets described below. The results for these different training regimes are cataloged in Table I.

### A. Datasets

*1) MNIST:* The original MNIST dataset [19] consists of a training set of 60,000 samples and a test set of 10,000 samples of handwritten digit images of size $28 \times 28$ grayscale pixels. As noted above, we used this dataset to pretrain the convolutional layers as part of a CNN classifier.

*2) Synthetic MNIST:* In order to approach the training of the models described in this paper, given the lack of accompanying motion trajectories, the MNIST dataset by itself does not provide the necessary image motion pairs for the data descriptions defined in Equations (1) or (3). Therefore, we employed the same scheme described in [1] in order to generate $40 \times 40$ as well as $60 \times 60$ images of synthetically written digits and associated two-dimensional artificial writing trajectory movements.[2] Briefly, the synthetic trajectory data was generated using a combination of straight lines and elliptic arcs. These geometric elements were used to generate grayscale digit images and their paramaters were varied according to a uniform distribution. The resulting images were processed with a Gaussian filter

[1]Code: `https://github.com/abr-ijs/imednet`
[2]Code: `https://github.com/abr-ijs/digit_generator`

TABLE I: DMP reconstruction statistics. The results are in pixels. The best result for each dataset is highlighted in boldface.

| | s-MNIST | s-MNIST-R | s-MNIST-RTS | s-MNIST-RTS-VW |
|---|---|---|---|---|
| IMEDNet | $\mathbf{0.1353 \pm 0.0491}$ | $0.3606 \pm 0.4746$ | $0.7745 \pm 0.0.4702$ | $0.8350 \pm 0.5117$ |
| CIMEDNet | $0.2285 \pm 0.0952$ | $0.6361 \pm 0.5329$ | $1.1785 \pm 0.4897$ | $1.2985 \pm 0.6104$ |
| STIMEDNet | $0.1750 \pm 0.0726$ | $\mathbf{0.3054 \pm 0.4911}$ | $\mathbf{0.2085 \pm 0.1172}$ | $\mathbf{0.2193 \pm 0.1851}$ |

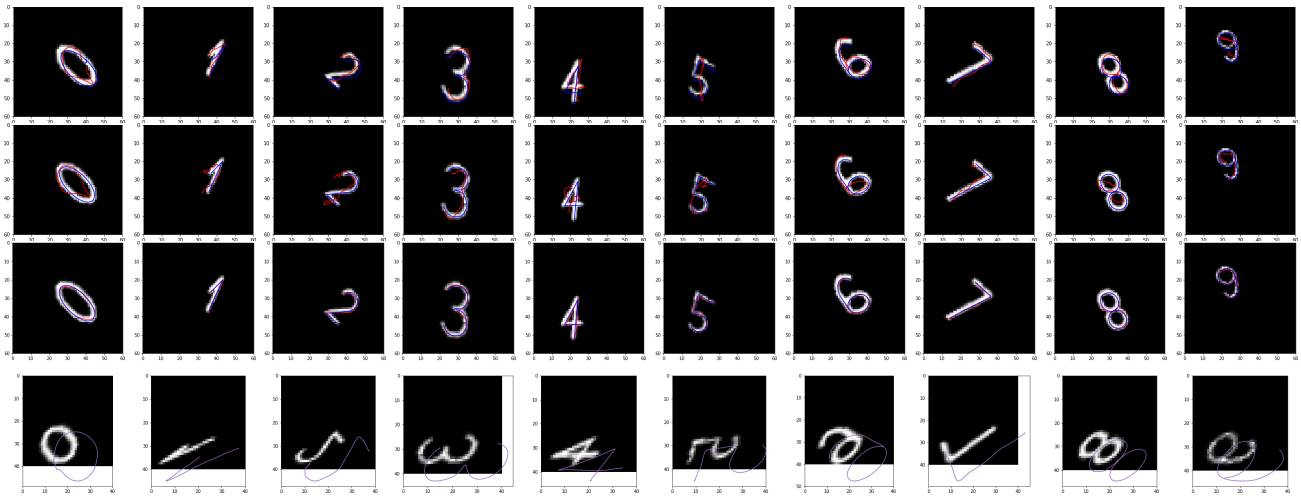

Fig. 4: Example results for three different models (IMEDNet in the top row, CIMEDNet in the second row, STIMEDNet in the third row and the canonical image and trajectory predicted by STIMEDNet in the bottom row) on the s-MNIST-RTS-VW dataset. The original trajectories are shown in blue, while the DMP trajectories calculated by the models are shown in red and the predicted canonical trajectory is shown in purple. None of the presented samples were used in training.

and some moderate salt-and-pepper noise was added to the foreground pixels. Finally, both the generated trajectories and the resulting images were transformed using affine transformations composed of translation, rotation, scaling, and shearing. These parameters were again taken from a uniform distribution. For the DMP representation of the trajectories, 25 radial-basis functions were selected for every dimension. The weights of these basis functions form together with the common time constant (1 parameter) and the start and the goal values of a planar movement ($2 \times 2$ parameters), the full set of 55 DMP parameters that represent the motion.

Using the above described procedure several datasets were generated both with and without similar affine distortions to those of some of the distorted MNIST datasets described in the spatial transformer networks paper [14] as follows:

- **s-MNIST:** 2000 pairs of images and trajectories without any added noise were generated for each digit, for a total of 20000 samples that were split in a $70\%/15\%/15\%$ ratio between training/validation/test data,
- **s-MNIST-R:** 2000 pairs of images and trajectories were generated for each digit as with s-MNIST, but the digits and trajectories were randomly rotated with uniform angle sampling between $-90^\circ$ and $+90^\circ$ as in [14],
- **s-MNIST-RTS:** 2000 pairs of images and trajectories were generated for each digit as with s-MNIST, but the digits and trajectories were randomly rotated with uniform angle sampling between $-45^\circ$ and $+45^\circ$ as in [14], scaled by a factor between $0.7$ and $1.2$ as in [14] and translated to a random location in a $60 \times 60$ image with a black background,
- **s-MNIST-RTS-VW:** 2000 pairs of images and trajectories were generated for each digit as with s-MNIST-RTS, but the line widths of the digits in the images were randomly varied within a standard deviation of $0.5$ of their original thicknesses.

It should be noted that the pre-trained CIMEDNet model in the STIMEDNet architecture was trained on the s-MNIST dataset described above. The main quantitative results are presented in Table I while qualitative results for selected samples are presented in Fig. 4. The quantitative evaluation presented in Table I uses dynamic time warping in order to measure the mean pointwise pixel distance between the trajectories generated by the DMPs predicted by the networks from the digit images and the actual digit trajectories. As can be seen from the results, STIMEDNet outperforms both the original fully-connected IMEDNet and the convolutional CIMEDNet in most cases, significantly so when the digits have undergone full affine transformations and it performs comparably to IMEDNet in the remaining case. The qualitative results from Fig. 4 also demonstrate how the STN finds canonical poses of the digits such that the image-to-motion encoder-decoder can predict canonical trajectories that can be

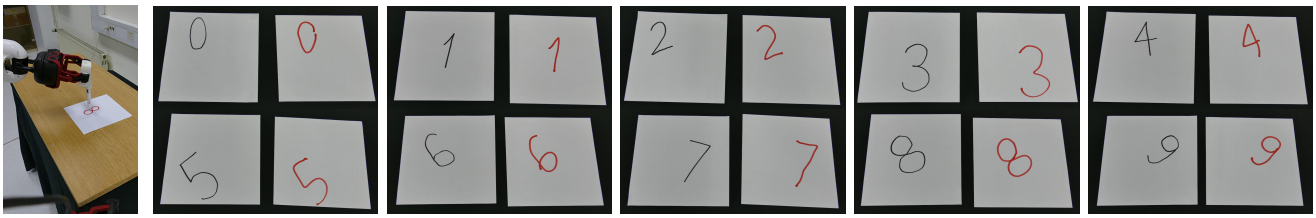

Fig. 5: Closeup of the Talos robot drawing a digit with its hand alongside results from the robot experiment. Digits handwritten by a human are shown in black ink, digits written by the robot are shown in red ink.

accurately transformed to the expected poses. These are not the kinds of canonical poses that might have been expected given the categorical nature of the digits in the dataset, or compared to what a human might have produced, but we reason that because this is a regression problem, as opposed to the classification problem considered by the original STN paper [14] there is no inherent bias in the objective function to produce canonical digit poses that are well-matched to their original categorical forms prior to distortion.

### B. Robot Experiment

In our experiment with the Talos humanoid robot, we used a trained STIMEDNet model to recognize handwritten digits written in arbitrary poses on pieces of paper placed on a whiteboard in front of the robot, and then to produce the DMPs and trajectories needed for the robot to draw the digits with its hand at matching locations on its own pieces of paper of the same size. In the first step, the pieces of paper with the digits written on them are segmented from the scene in the robot's camera image using standard computer vision techniques as follows. The whole image from the robot camera scene view is blurred and transformed into HSV color space, where we use a Canny edge detector in the value dimension. In the extracted edges, we search for closed contours with the shape of a parallelogram and appropriate sizes. The content of appropriate contour is then affine transformed to a square shape and forms the image of the piece of paper with the handwritten digit resized to the 60x60 pixels required for input into the neural network. This image is further processed to gray color, normalized and color inverted. For lowering the influence of light and more closely matching the intensity of the database images, the images of the handwritten digits are thresholded. In order to more closely match the width of the pen used by the human writer to the width of the digits in the training database, we dilated the images with kernel size 2 and finally applied some light Gaussian filter smoothing.

The processed images are fed to the neural network which returns a trajectory for writing the digits from the images. Since the trajectories from the neural network are represented in image space, they are first transformed into the task space coordinate system of the robot before being executed with the robot arm. The trajectory from the neural network defines the planar movement in two dimensions: the height and orientations are held constant throughout all of the trajectory executions. The robotic arm uses position control and the desired force of the pen on the paper is ensured with a spring placed behind the pen in the pen holder. The results are shown in Fig. 5. As can be seen, this proof-of-concept shows that the robot can use the STIMEDNet network to learn to both read and write digits in arbitrary affine poses[3].

## IV. CONCLUSIONS AND FUTURE WORK

We have presented a novel neural network architecture for image-to-motion prediction that employs a spatial transformer module, a convolutional image-to-motion encoder-decoder module and a motion transformer module in a fully-differentiable overall model. We have demonstrated that this architecture outperforms its predecessor on a variety of different datasets and we have demonstrated its use with a full-scale humanoid robot that is able to use the network to learn to read and write digits in arbitrary poses as presented to it. Regarding future work, we intend to expand the capabilities of these models still further by incorporating layers from more powerful pre-trained CNN models into the image encoder of the CIMEDNet part of the network and training the network on more challenging image sets. One challenge here lies in either finding suitable image datasets that include trajectory information in their target outputs or in finding ways to convert the data in existing datasets into trajectories. A possible approach would be to convert the borders of object labels in existing semantic segmentation datasets into draw trajectories, however, this would present the additional difficulty of having to account for DMP representations for trajectories that vary significantly in length given the large variation in object size in such datasets. This might be resolved by employing a recurrent neural network (RNN) architecture and using fixed-sized DMPs in a temporal piecewise construction scheme. An interesting extension to the original spatial transformer network that includes an RNN capability was proposed in [20] and we envisage using a similar technique within our architecture.

**Acknowledgement**: This work has received funding from the EU's Horizon 2020 RIA AUTOWARE (GA no. 723909); the Slovenian Research Agency under GA no. J2-7360; JSPS KAKENHI JP16H06565; NEDO; the Commissioned Research of NICT; the NICT Japan Trust (International research cooperation program); and was supported by JST-Mirai Program Grant Number JPMJMI18B8, Japan. The authors also wish to thank Marcel Salmič for his significant contribution to the PyTorch network implementations.

[3]Video: `https://youtu.be/21YQd6_h8QE`

## REFERENCES


[1] R. Pahič, A. Gams, A. Ude, and J. Morimoto, "Deep Encoder-Decoder Networks for Mapping Raw Images to Dynamic Movement Primitives," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, May 2018, pp. 5863–5868.

[2] 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.

[3] T. S. Cohen and M. Welling, "Transformation Properties of Learned Visual Representations," in *International Conference on Learning Representations (ICLR)*, San Diego, 2015.

[4] K. Lenc and A. Vedaldi, "Understanding Image Representations by Measuring Their Equivariance and Equivalence," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 991–999.

[5] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic Routing Between Capsules," in *Advances in Neural Information Processing Systems (NIPS) 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 3856–3866.

[6] A. Graves, "Generating Sequences With Recurrent Neural Networks," *arXiv:1308.0850 [cs]*, Aug. 2013.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra, "DRAW: A Recurrent Neural Network For Image Generation," in *International Conference on Machine Learning*, Jun. 2015, pp. 1462–1471.

[9] D. Ha and D. Eck, "A Neural Representation of Sketch Drawings," *arXiv:1704.03477 [cs, stat]*, Apr. 2017.

[10] O. Ali, "Robotic Calligraphy: Learning From Character Images," Ph.D. dissertation, 2015.

[11] P. C. Yang, K. Sasaki, K. Suzuki, K. Kase, S. Sugano, and T. Ogata, "Repeatable Folding Task by Humanoid Robot Worker Using Deep Learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 397–403, Apr. 2017.

[12] A. Pervez, Y. Mao, and D. Lee, "Learning deep movement primitives using convolutional neural networks," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, Nov. 2017, pp. 191–197.

[13] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-Specific Generalization of Discrete and Periodic Dynamic Movement Primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, Oct. 2010.

[14] M. Jaderberg, K. Simonyan, A. Zisserman, and k. kavukcuoglu, "Spatial Transformer Networks," 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., 2015, pp. 2017–2025.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

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

[17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS 2017 Autodiff Workshop*, Oct. 2017.

[18] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference for Learning Representations (ICLR)*, San Diego, 2015.

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

[20] S. K. Sønderby, C. K. Sønderby, L. Maaløe, and O. Winther, "Recurrent Spatial Transformer Networks," *arXiv:1509.05329 [cs]*, Sep. 2015.