2 Supplementary: Selection for models and their parameters

2.1 Encoder

2.1.1 Comparison with the different feature extractors

The encoder transforms input signals into a distributed output, and the output is phase-free. An inherent characteristic of this approach is that the output with similar trait tends to group nearer to one another. Using the objective criteria

$$\boldsymbol{\theta} = \underset{\boldsymbol{\theta}}{argmin}\left(\frac{e(\boldsymbol{\theta})}{d(\boldsymbol{\theta})}\right) \tag{2.1}$$

and

$$\boldsymbol{\theta} = \underset{\boldsymbol{\theta}}{argmin}\left(\frac{e(\boldsymbol{\theta})}{d(\boldsymbol{\theta})} + MSE\left(g_{e_t}^{-1}\left(\boldsymbol{h}_t^{(2)};\boldsymbol{\theta}\right), \boldsymbol{h}_{(t-c+1):(t)}^{(1)}\right) + \zeta\left(\boldsymbol{h}_t^{(3)}\right)\right), \tag{2.2}$$

the aim of this experiment is to find a good encoder model from a list of feature extractors.

The five outputs are the Raw, FFT, FFT2, CNN, and K-mean. The Raw is spatial and distributed. This output is acquired by transforming the upper-limb data with a spatial gesture encoder. The parameters of the spatial encoder were trained using contrastive divergence. The FFT and FFT2 are the 1D and 2D Fourier transformation of the upper-limb data. For both methods, the inputs are either 75 frames (15 FPS) or 25 frames (5 FPS) of data. Furthermore, only a part of the transformation was extracted. The FFT holds the amplitude of the first ten Fourier components for each feature. On the other hand, the FFT2 stores the amplitude of the first ten Fourier components in the time domain and all others in the spatial domain. In the end, both the output will have the same number of features, 180, for each time stamp. The CNN is a mean pooled network with its features trained by RBM or Autoencoder. For the paper, the T window has 25 frames (five seconds), c window has five frames (one second), and the number of convoluted output is 200. Lastly, the K-mean has its local features trained using fuzzy K-mean, and mean pooling is the pooling strategy. Like the CNN, the K-mean has a T window and c window of 25 frames (five seconds) and five frames (one seconds) respectively. The number of clusters is 200.

Figure 2-2 displays the distributions of different gestures styles. The first observation is that the distributions are overlapping one another (Figure 2-2). Furthermore, the Raw also produces distributions with multi-peaks within a single class. Figure 2-1 (top) shows one of the class from the Raw with multi-peaks. On the contrary, Figure 2-1 (bottom) displays that the CNN creates a distribution with a peak. Multi-peaks in a class is an undesirable distribution because it implies that there is more than one cluster for that class. The last observation is that there is coincided peak in both the Raw and CNN. For the Raw, the coincided peaks occur in both the principal components (Top of Figure 2-1). For the CNN, the coincided peaks only occur on one of the principal components (PCA1 of Figure 2-1 bottom). In short, the visual observation shows the advantage of the spatiotemporal over the spatial output.

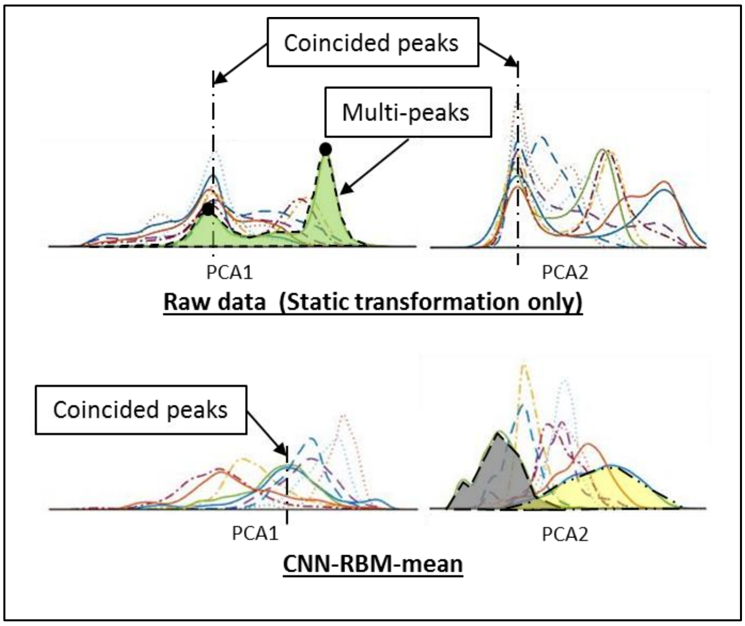

Figure 2-1 The top figure shows the location of the coincided peaks of multiple classes and the multi-peaks for a specific class for the raw data. The bottom figure shows the histogram of CNN with the coincided peaks of two specific classes at PCA1, and the same classes being separated in PCA2.

From Table 2-1, the CNN200-RBM-Mean is the selected model for the encoder. Among other neural approaches that use (2.2), the CNN200-RBM-Mean is the second best. Both CNN200-RBM-Mean and CNN200-Autoencoder-Mean have a very score of 1.770(0.001) and 1.769(0.001) respectively. The t-test gives a P-value of 0.1525, which is not significant. However, the CNN200-RBM-Mean has a lower score with (2.1). The CNN200-RBM-Mean scores about 0.776(0.001), while the CNN200-Autoencoder-Mean scores 0.799(0.001). The t-test gives a P-value less than 0.0001, which is significant. In term of (2.2), the neural networks with the spatial layer scored lower compare to the neural networks without this layer. The RBM and Autoencoder without the spatial layer score 1.770(0.001) and 1.769(0.001) respectively. These values are lower than their counterparts with spatial layer, which score 1.781(0.007) and 1.775(0.000) respectively.

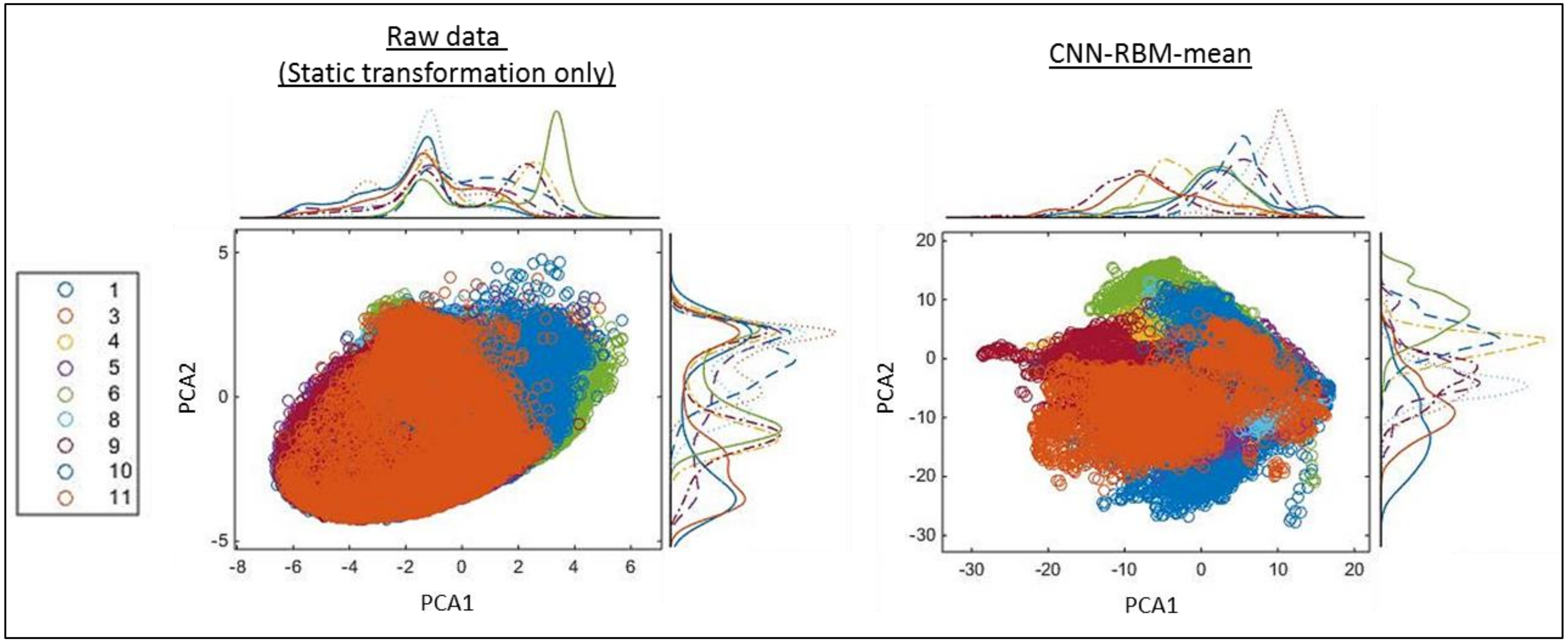

Figure 2-2 Distribution of classes in their new vector space: Spatial transformation and CNN. Each distribution comprises of two histograms and a 2D scatter plot. The two histograms estimate the class distribution for the data on the first and second PCA components, which the x-axis and y-axis respectively.

Table 2-1 Comparison between the Raw and the various encoded features with the objective functions. The CNN and K-mean have their average distances computed from five different training data set.

Objective Function	Raw	FFT (15/5 FPS)	FFT2 (15/5 FPS)	CNN200-RBM-Mean	CNN200-RBM-Mean-Spatial	CNN200-Autoencoder-Mean	CNN200-Autoencoder-Mean-Spatial	Fuzzy-K-mean
(2.1)	1.578	1.307 / 1.394	1.387 / 1.396	**0.776(0.001)**	0.782(0.000)	0.799(0.001)	0.799(0.001)	1.430(0.022)
(2.2)	--	--	--	**1.770(0.001)**	1.781(0.007)	1.769(0.001)	1.775 (0.000)	--

2.1.2 Selection of the parameters

There are many hyperparameters for a CNN model. The hyperparameters include number of the neuron, type of the neuron (e.g. rectified or binary neuron), sparsity, dropout, pooling function, and many more. Due to many parameters, this experiment only shows the results for some selected parameters. The aim of the analysis is to find the best configuration across three factors, and they are the number of the neurons, decay rate constant, and types of decay.

The descriptions of the three factors are as follows. The number of the neurons is the number of convoluted or spatiotemporal features. A higher number of neurons increases the performance, but it will overfit towards the training data. Hence, regulator, sparsity, and dropout are some of the tools to generalize the model. As mentioned, the regulator is one of the tools to generalize the model. The regulator periodically decays the weight by a decay rate, and this prevents any feature from dominating. The L1-norm and L2-norm are the two different ways of regulating the weights. They have their own properties. The L2-norm produces more distributed features, while L1-norm produces a sparser weight.

Table 2-2 Test result using (2.2) for CNN-mean with rectified dropout neuron and Adadelta learning rate. The rows represent the number of neurons, and the columns store two different parameters: the type of decay and the decay rate.

		Regulator Parameters			
		L1-norm		L2-norm	
		1e-5	1e-4	1e-5	1e-4
Number of Neurons	50	1.808(0.001)	1.815(0.001)	1.810(0.001)	1.810(0.000)
	100	1.783(0.002)	1.788(0.003)	1.785(0.001)	1.785(0.001)
	200	**1.770(0.001)**	1.779(0.001)	1.771(0.002)	1.772(0.002)

From the Table 2-2, the best performance comes from a CNN with 200 neurons, 1e-5 decay rate, and L1-norm. A higher number of neurons improves the performance of the model. The L2-norm produces consistent results given the two decay rates; on the contrary, the L1-norm results have a higher difference between the two decay rates. However, the L1-norm yields a better performance against the L2-norm when the decay rate is 1e-5 at all number of neurons. As a result, CNN with 200 neurons, 1e-5 decay rate, and L1-norm is the chosen CNN for the future experiment.

2.2 Decoder

2.2.1 Comparison of the LSTM with the different feature extractors

One of the goals of this experiment is to check whether the result at Section 2.1.1 is corrected. The next goal is to examine the decoder performance as a gesture generator. From Table 2-3, the CNN-RBM-Mean gives the best result for the LSTM, which matches the encoder results. With a mean score of 0.257 for DTWMSE, the CNN-RBM-Mean has a higher performance of 17.628% when compared with the Fuzzy-K-mean, which is in second place. In term of the MSE score, the CNN-RBM-Mean also has the lowest score. However, there is an inequality between the MSE and DTWMSE. This difference implies that the synthesize gestures spatially match the actual gestures. However, the cycle time might be off by a maximum of five frames, which correspond to the DTW local constraint w.

Table 2-3 Comparison between different encoded features with gate size of 300 and L2-norm. MSE is the mean square error of the reconstructed signals with respect to the actual signals, while the DTWMSE is the mean square error of the realigned reconstructed signals with respect to the actual signals.

	FFT	FFT2	CNN-RBM-Mean	Fuzzy-K-mean
BPTT Ratio	0.3	0.3	0.2	0.3
Lambda	1e-6	5e-6	5e-6	1e-6
MSE	0.547(0.004)	0.629(0.002)	**0.473(0.004)**	0.528(0.003)
DTWMSE	0.342(0.002)	0.401(0.004)	**0.257(0.004)**	0.312(0.002)

2.2.2 Selection of the parameters and the effects of high neuron size

This section has two results, and they show the effect of various factors affecting the decoder training strategy. Figure 2-3 displays the DTWMSE results from four factors, which give 24 decoder parameters. The factors are the connecting internal state $\tilde{\boldsymbol{y}}_t$, the size of the gate N^{gate}, weight decay constant λ, and type of backpropagation γ. The connecting internal state is the set of internal state pass to the next time step as input. The size of the gate governs the number of elements in all the gate and cell. The weight decay constant regulates the model by periodically reducing the weight. Lastly, the type of back propagation alternates the training strategies from truncated BPTT to ratio BPTT. Table 2-4 gives the relationship between the parameter size and the DTWMSE result. There are three columns in the table. Each column has the parameter size from each weight, not including bias. The parameter size is calculated by multiplying the input to output, where the input includes the five earlier upper-limb states, the encoded data, and earlier LSTM internal state. Each column also holds the total parameter size and its difference in comparison to the last configuration. Lastly, it also has the DTWMSE result and its changes in relation to the last column.

From Figure 2-3, the best configuration is (2, 2, 2, 2) where $\tilde{\boldsymbol{y}}_t = \boldsymbol{y}_{cell,t}$, $N^{gate} = 300$, $\lambda = 5e^{-6}$, and $\gamma = 0.2$. The connecting internal state $\tilde{\boldsymbol{y}}_t \in \{\emptyset, \boldsymbol{y}_{cell,t}\}$ plays a crucial role in bringing down the median and ranges. Based on the t-test result on the internal state, the P-value is >0.0000, which is significant. Given $\tilde{\boldsymbol{y}}_t = \boldsymbol{y}_{cell,t}$, the size of the gate N^{gate} gives an added push downwards. However, this effect is not as significant when $\tilde{\boldsymbol{y}}_t = \emptyset$. The P-value on the size of gate given $\tilde{\boldsymbol{y}}_t = \boldsymbol{y}_{cell,t}$ is >0.0000, which is significant. On the other hand, the P-value is 0.0242 when the t-test only based on the size of the gate. As of remaining factors, there is no consistency significance between their values.

Increasing parameters improves the performance of the model is a well-known concept. However, increasing the size of the parameter might not be effective to improve the performance, and Table 2-4 verified the statement. Using $\tilde{\boldsymbol{y}}_t = \emptyset$ and $N^{gate} = 300$ as the benchmark, the performance increases by 7.24% and the size of parameter grows by 11.94% when there is more input $\tilde{\boldsymbol{y}}_t = \boldsymbol{y}_{cell,t}$ and the gate size $N^{gate} = 200$. It is equivalent to 1.629% increase of the parameter size to gain 1% of accuracy. However, the performance deteriorates when $\widetilde{\boldsymbol{y}}_t = \boldsymbol{y}_{cell,t}$ and $N^{gate} = 300$. The efficiency decreases to 8.951% of parameter expansion to yield 1% of accuracy. There is an increase in parameter expansion by 5.494 times to yields the same result. Hence, the above experiments stop investigating on adding more input or increasing the gate size because higher parameter size will affect the computation time during runtime.

2.3 Associator

2.3.1 Selection of the parameters

The aim of the trial is to find the best configuration the associator, and there is a total of three factors. The factors are vigilance thresholds ρ, pruning threshold τ, and learning rate η. Firstly, the vigilance threshold is the parameter to toggle between fast and slow learning. For this paper, there are two of them, and they are for the gesture and facial modalities. The range of this thresholds is from 0.7 to 0.9 with an interval of 0.1. Next, the pruning threshold is to remove any outlier rules, which is below a particular frequency. It was set at no pruning, 0.04, and 0.05. Lastly, the learning rate is the amount of plasticity in the knowledge during slow learning. It is set at adaptive, 0.05, 0.1, and 0.2. In short, the aim of the experiment is to find the best parameters that give a good accuracy with the least amount of weights.

There are a few observations from this experiment. First, the pruning function reduces a large amount of weight. From Table 2-6, the pruning function reduces the total number of weight from >40 to <10. However, the amount of weight does not diminish as much with the increase of the pruning threshold. Secondly, there might have a crossover interaction between learning and pruning threshold. From Table 2-5, no pruning threshold and adaptive learning rate yield 0.092 for DTWMSE. The DTWMSE is 0.106 when the learning rate $\eta = 0.2$. The trend was not significant when the pruning threshold τ = 0.01, and the trend was opposite when the pruning threshold τ = 0.05. Lastly, the best parameters are adaptive learning rate, pruning threshold at 0.04, face vigilance threshold is 0.9, and gesture vigilance threshold is 0.7. The parameters with the second best DTWMSE was selected because its number of weight is seven times lesser than the best DTWMSE.

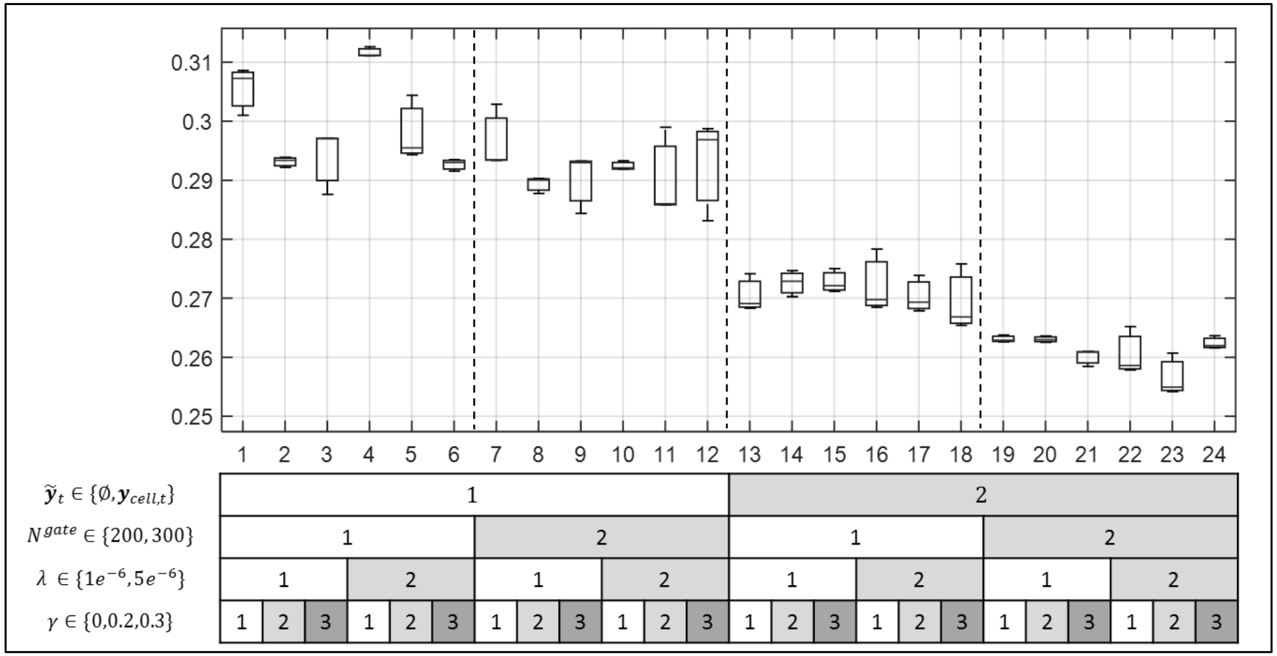

Figure 2-3 The DTWMSE results gathered from a permutation of the four factors with three replicas.

Table 2-4 The relationship between the parameter size respects with to the DTWMSE result

	$\tilde{\boldsymbol{y}}_t = \boldsymbol{y}_{cell,t}$ and $N^{gate} = 300$	Size of $\boldsymbol{W_i}$	$\tilde{\boldsymbol{y}}_t = \boldsymbol{y}_{cell,t}$ and $N^{gate} = 200$	Size of $\boldsymbol{W_i}$	$\tilde{\boldsymbol{y}}_t = \emptyset$ and $N^{gate} = 300$	Size of $\boldsymbol{W_i}$
$\boldsymbol{W}_{in,g_{in}g_{for}g_{out}}$	(90+200+300) * 300	177,000	(90+200+200) * 200	98,000	(90+200) * 300	87,000
$\boldsymbol{W}_{out}$	300 * 18	5,400	200 * 18	3,600	300 * 18	5,400
Total Parameters		713,400		395,600		353,400
Ratio		**↑101.87%**		**↑11.94%**		-
DTWMSE		0.257(0.003)		0.269(0.006)		0.290(0.008)
Ratio		**↑11.38%**		**↑7.24%**		-

Table 2-5 The median and standard deviation of the best DTWMSE results with respect to its learning rates and pruning threshold given the vigilance thresholds at Table 2-7. The rows of tables represent the learning rates, and the columns of the tables denote the different pruning thresholds

	$\tau = 0.00$	$\tau = 0.04$	$\tau = 0.05$
$\eta(\varphi_J)$	*0.092(0.051)* $_1$	**0.096(0.048)** $_2$	0.109(0.046)
$\eta = 0.05$	0.103(0.058)	0.101 (0.049)	0.106(0.042)
$\eta = 0.1$	0.102 (0.053)	0.103(0.046)	0.103(0.046)
$\eta = 0.2$	0.106(0.053)	0.101(0.047)	0.099(0.036)

Table 2-6 Total number of weights after pruning with respects to the DTWMSE result in Table 2-5

	$\tau = 0.00$	$\tau = 0.04$	$\tau = 0.05$
$\eta(\varphi_J)$	*49(10.2881)*	**7(1.3211)**	6(1.5395)
$\eta = 0.05$	223(34.0363)	6(1.8141)	5(1.5725)
$\eta = 0.1$	221(34.2489)	5(1.385)	5(1.1015)
$\eta = 0.2$	56(12.1873)	5(1.5507)	4(1.461)

Table 2-7 The vigilance thresholds with the best DTW_RMS result at a specific learning rate and pruning threshold.

	$\tau = 0.00$	$\tau = 0.04$	$\tau = 0.05$
	$\rho^{(f_*)}$ $\rho^{(g_{user})}$	$\rho^{(f_*)}$ $\rho^{(g_{user})}$	$\rho^{(f_*)}$ $\rho^{(g_{user})}$
$\eta(\varphi_J)$	0.7 0.8	**0.9 0.7**	0.8 0.7
$\eta = 0.05$	0.9 0.9	0.8 0.7	0.7 0.7
$\eta = 0.1$	0.8 0.9	0.7 0.7	0.7 0.7
$\eta = 0.2$	0.7 0.8	0.9 0.7	0.9 0.7