Metamorphic Forward Adaptation Network: Dynamically Adaptive and Modular Multi-layer Learning

Anonymous authors
Paper under double-blind review

Abstract

Back-propagation is a widely used algorithm for training neural networks by adjusting weights based on error gradients. However, back-propagation is gradient-dependent, biologically implausible with global derivative computations, and lacks robustness in long-term dynamic learning. A promising alternative to back-propagation is the Forward-Forward algorithm, which bypasses gradient dependency and localizes computations, making it a more biologically plausible approach. However, Forward-Forward has been evaluated in limited environments, not yet matched back-propagation's performance, and only supports classification, not regression. This research introduces the Metamorphic Forward Adaptation Network (MFAN), using contrastive learning property as its core, and retaining the layer-wise architecture of the Forward-Forward algorithm. Compared to the Forward-Forward model being limited to discrete classification, MFAN can process both discrete and continuous data, also showing stability, adaptability, and ability to handle evolving data. MFAN performs well in continual learning scenarios, demonstrating superior adaptability and robustness compared to back-propagation and Forward-Forward approaches, particularly in tasks requiring dynamic, long-term learning.

1 Introduction

Deep learning has achieved remarkable success across numerous applications, largely due to the efficiency of the back-propagation (BP) algorithm in training deep neural networks. BP has been the backbone of most advancements in the fields from image recognition to natural language processing. However, as these models are increasingly deployed in dynamic environments – such as adaptive systems and online continual learning – BP's limitations become more pronounced.

One shortcoming of BP is its limitations in real-time applications which require maintaining the plasticity Dohare et al. (2024). BP often requires precise knowledge of the computations in the forward pass to calculate accurate gradients, making it less suited for scenarios where the underlying data distributions are continuously changing. Additionally, BP typically processes batch data, requiring the accumulation of data before updating weights, which is not ideal for real-time or continual learning tasks. As BP relies on global gradient updates, BP-trained models often struggle to adapt to dynamic environments or generalize to evolving tasks

As a solution, adaptive neural networks Ashfahani & Pratama (2019) – which can dynamically modify their architecture by expanding or contracting based on task complexity – have shown better performance in real-time settings. For example, tasks like continuously streaming data require immediate adaptation and stable predictions. In such scenarios, adaptive models have been proven more effective.

A point of contention regarding BP is its biological implausibility Bengio et al. (2015). There is little evidence for the backward flow of error signals, which is central to BP, in biological neural systems. Neurons in the brain likely operate based on local updates rather than the global error propagation seen in BP. This disconnect has fueled interest in finding more alternative learning methods Karimi et al. (2024); Lv et al. (2024), such as the Forward-Forward (FF) algorithm Hinton (2022).

The FF algorithm is a greedy multi-layer learning procedure inspired by Boltzmann machines Hinton et al. (1986) and Noise Contrastive Estimation Gutmann & Hyvärinen (2010). FF replaces BP's forward-backward passes with two forward passes – one with positive (real) data and one with negative (generated or corrupted) data. Each layer of FF has its local objective function aiming to have high goodness for positive data and low goodness for negative data through weight adjustments in every hidden layer (Fig. 1a).

One example of such a goodness measure is the sum of the squares of the activities of the rectified linear neurons in each layer, aiming to learn to make the positive data goodness above a threshold θ and the negative data below. Thus FF can classify input vectors by computing the probability that an input vector is positive. This is achieved by applying the logistic function, σ to the goodness minus θ,

$$p(positive) = \sigma\left(\sum_j y_j^2 - \theta\right) \tag{1}$$

where y_j is the activity of hidden unit j before layer normalization.

While BP requires a fully differentiable forward process to propagate learning signals throughout the network, FF can free this constraint by having each layer learn with locally generated loss signals.

Even though FF provides a more modular, layer-wise learning approach, it tends to be slower, particularly when applied to large-scale tasks, and its reliance on local optimization can result in sub-optimal coordination across layers, potentially reducing overall performance in larger networks.

Researchers have explored several enhancements to the FF algorithm, including hyper-parameter optimization and hybrid approaches Gandhi et al. (2023); Izzo et al. (2024); Kumar et al. (2023); Ahamed et al. (2023); Reyes-Angulo & Paheding (2024); Wang et al. (2024). One study Gandhi et al. (2023) introduced a pyramidal strategy for adjusting the loss threshold progressively across layers, enhancing performance on tasks beyond MNIST classification, including sentiment analysis on the IMDb dataset. Additionally, integrating the FF algorithm with BP Izzo et al. (2024) has shown promising results, combining the robustness of BP with the biological plausibility of FF. These hybrid models have demonstrated improved noise resilience and adaptability, especially in environments where task complexity evolves.

In light of the efforts and limitations outlined above, our research identifies an additional constraint of the FF model that has yet to be addressed. The FF model's design combines both x and y as inputs. During inference, this requires knowledge of all possible discrete y values to combine with the test x to create valid test inputs. The model then computes goodness values for these inputs and selects the y prediction corresponding to the highest goodness value. While this design facilitates classification, it poses challenges for regression tasks.

To address these limitations, this paper proposes the Metamorphic Forward Adaptive Network (MFAN), a novel architecture using contrastive learning property as its core, and utilizing a layer-wise architecture similar to FF. MFAN extends FF's algorithm to the continuous data domain, supporting both regression and classification.

The first difference in the design of MFAN and the FF algorithm is data processing. Negative y (y_{neg}) is generated from positive y (y_{pos}) by either choosing the wrong label (classification) or introducing random noise offsets (regression). MFAN no longer uses combined x and y_{neg}/y_{pos} as input, and instead separately encodes them into the same d-dimensional latent space with an input encoder f_x, and an output encoder f_y, obtaining,

$$z_x = f_x(x) \tag{2}$$

$$z_{y_{pos}} = f_y(y_{pos}) \tag{3}$$

$$z_{y_{neg}} = f_y(y_{neg}) \tag{4}$$

where $z_x, z_{y_{pos}}, z_{y_{neg}} \in \mathbb{R}^d$.

Then replacing FF's goodness measure, each MFAN layer independently optimizes a contrastive loss using cosine similarity Lahitani et al. (2016). Cosine similarity is a metric used to measure the similarity between

two vectors by calculating the cosine of the angle between them,

$$cos\_sim(v_1, v_2) = \frac{v_1 \cdot v_2}{\|v_1\|\|v_2\|}. \tag{5}$$

As cosine similarity measures the alignment between vectors, the contrastive loss function uses this property to encourage MFAN to learn to pull z_x and $z_{y_{pos}}$ together and push z_x and $z_{y_{neg}}$ apart in latent space $\mathbb{R}^d$,

$$Loss_{cos\_sim} = -\frac{z_x \cdot z_{y_{pos}}}{\|z_x\|\|z_{y_{pos}}\|} + \alpha_1 \frac{z_x \cdot z_{y_{neg}}}{\|z_x\|\|z_{y_{neg}}\|} + \alpha_2 \frac{z_{y_{pos}} \cdot z_{y_{neg}}}{\|z_{y_{pos}}\|\|z_{y_{neg}}\|} \tag{6}$$

where α_1 and α_2 are tunable parameters, the last term of this loss aims to retain the distance between positive and negative y data mappings. Fig. 1d is a 2D illustrative diagram of this loss.

Using this loss function, MFAN encoders are first trained together to obtain mappings from the input/output space to the latent space, pulling $z_x \simeq zy_{pos}$. The encoder parameters are then frozen and used to obtain (y, z_y) pairs. These are used to train a decoder f_y^{-1}, which learns the reverse mapping of $z_{y_{pos}}$ back to y_{pos} (Fig. 1e), assuming $z_x \simeq zy_{pos}$

$$\hat{y}_{pos} = f_y^{-1}(z_x) = f_y^{-1} f_x(x). \tag{7}$$

At inference time, the input encoder and decoder are together used to make predictions,

$$\hat{y}_{test} = f_y^{-1} f_x(x_{test})). \tag{8}$$

This allows MFAN to predict without knowing all possible y, hence being able to process continuous data and do regression tasks.

Moreover, since each layer makes a prediction, an effective layer selector is used to select the layer that will be used to make the final prediction. When doing classification tasks, the layer with the best performance on the training set is used at inference time. One potential selection criterion is that the model can record several top performance layers during training and take their outputs' average at inference time as the prediction to generate smoother predictions and maximize stability. Here, a straightforward strategy is employed for the regression task. MFAN selects the index of the layer with the best performance during training (e.g. L_2) and uses the output from a layer two levels deeper (e.g. L_4) during inference. This strategy choice is guided primarily by empirical validation results demonstrating its effectiveness and may be further justified by the greater task complexity during inference, where the need for extrapolation benefits from deeper layer analysis. For online continual learning tasks, the model records the best-performing layer at the previous time step, and at the subsequent evaluation time, the model uses that layer's prediction. This layer selector effectively allows the model to adjust the network depth dynamically.

2 Experiments and Evaluations

2.1 MNIST Classification: Performance on Digit Classification

The MNIST dataset was used to validate MFAN on a classification task and compare its performance to a basic BP baseline model. The setup of both the BP and MFAN was designed to ensure comparability between the two rather than optimizing for peak performance.

The learned latent space for each layer in MFAN is visualized using t-distributed stochastic neighbor embedding (T-SNE) (Fig. 1c). The data points in the latent space are semantically grouped by their labels. The data points form clusters that correspond to each label in the latent space learned by each layer's encoder. This shows that the model learns semantically meaningful features.

With similar model configurations (see Table A.1), MFAN achieved an average test accuracy of $91.65\% \pm 0.20\%$, while the BP model reached $94.36\% \pm 0.46\%$. Although MFAN demonstrated more consistent performance with half the standard deviation, it lagged behind BP by 2.71% in accuracy, indicating that while MFAN is competitive, it does not yet match the BP model's performance on this computer vision task.

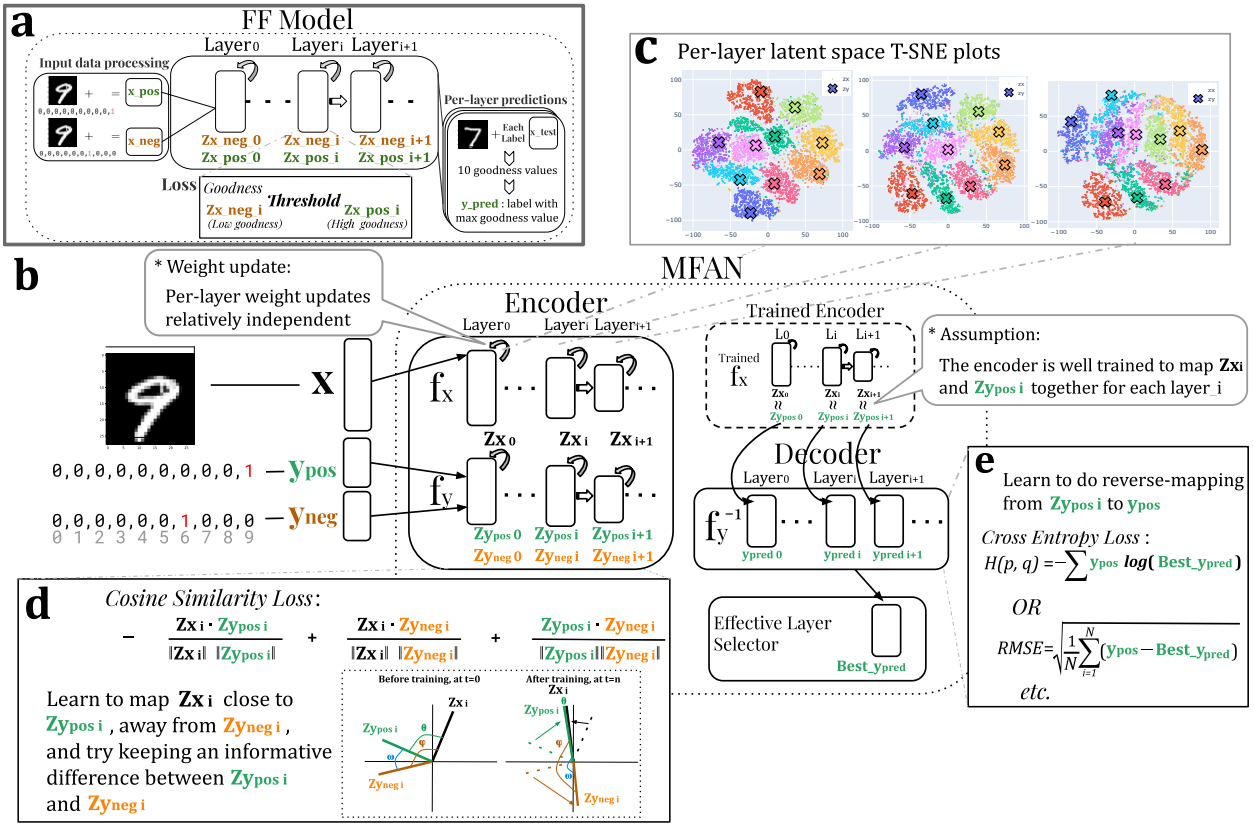

Figure 1: **a)** Schematic for the Forward-Forward (FF) model. The FF model combines x data and y labels into "good" and "bad" data inputs. Each layer has its own loss function that maximizes/minimizes goodness for 'good'/'bad' data. At inference the input is combined with all potential labels and the model is used to measure goodness for each. The label with the highest goodness is taken as the output. This can not be applied to continuous data, such as regression. **b)** Schematic for the proposed Metamorphic Forward Adaptive Network (MFAN). MFAN consists of an input encoder f_x, output encoder f_y and a decoder f_y^{-1}. The encoders map the input data x and its corresponding output y into the same latent space: $z_x = f_x(x)$, $z_y = f_y(y)$ and the contrastive loss is used to pull positive samples of x and y together and push negative samples apart. The decoder is used to map from the latent space to the predicted output $\hat{y} = f_y^{-1}(z)$. The effective layer selector is used to select the layer that will be used to make the final prediction. **(c)** Latent space embedding for each MFAN layer in MNIST classification tasks. **(d)** Cosine similarity is used for the contrastive learning loss.

It is worth noticing that the MFAN layer selector primarily selects the second-layer output instead of the final layer (test accuracy $90.12\% \pm 0.28\%$). This dynamic reduction in model depth helps MFAN mitigate over-fitting to the training data at inference time. Conversely, it is reasonable to suggest that the layer selector can prevent under-fitting by extending the network to deeper layers when necessary, as demonstrated in the upcoming function regression evaluations.

2.2 Function Regression: Continuous Data Learning

We evaluate MFAN on continuous data using a series of trigonometric functions as regression targets Hay & Sharon (2024); Xu et al. (2021). We use a three-layer BP model as the baseline for comparison. To demonstrate the adaptive layer selection mechanism of MFAN, we implement a six-layer MFAN architecture. This configuration enables MFAN to dynamically select from a broader range of layers, allowing it to utilize deeper layers beyond the three layers of the BP model or to opt for shallower layers according to the specific requirements of the tasks. The detailed evaluation mechanism is explained in Fig. 2.

Observations show that MFAN dynamically chooses output layers during inference, adjusting the effective network depth as needed. This shows that MFAN has more prediction choices compared to BP which has only one result. With these choices, the effective layer selector of MFAN helps decide the layer to use at inference time, as it records layers' performances during training and uses them as guidance. In contrast, BP produces less accurate or smooth prediction plots, likely due to its rigid network structure with fixed depth, unable to adapt to varied task complexity.

Table. 1 displays the average percentage error results of these function regression tasks. MFAN has much lower standard deviation values and has not been observed to fail any regression tasks in this experiment. The average percentage root mean square errors (RMSE) of MFAN are also lower than BP. These results demonstrated that MFAN consistently outperformed BP in stability and accuracy, emphasizing its ability to handle complex, nonlinear functional relationships and extrapolate effectively to unseen domains.

These findings highlight MFAN's robustness in handling continuous data, particularly in scenarios where the relationship between input and output variables is complex and non-linear. The combination of dynamic layer depth selection, contrastive learning principles, and layer-wise weight updates makes MFAN a more powerful tool for function regression tasks, particularly when stability and accuracy are important.

Table 1: Regression results: Percentage mean ± standard deviation

Regression on $f(x)$	MFAN[2] error	BP error
$f(x) = sin(0.2 * x)$	$06.43\% \pm 0.22\%$	$10.59\% \pm 04.02\%$[1]
$f(x) = cos(0.2 * x)$	$06.41\% \pm 0.35\%$	$21.16\% \pm 10.11\%$[1]
$f(x) = sin(0.4 * cos(0.2 * x)))$	$07.09\% \pm 1.09\%$	$19.63\% \pm 12.13\%$[1]
$f(x) = sin(0.4 * cos(0.2 * x) + 30))$	$10.39\% \pm 0.07\%$	$13.48\% \pm 00.56\%$
$f(x) = sin(0.8 * cos(0.2 * x) + 20)$	$09.03\% \pm 0.11\%$	$18.51\% \pm 11.68\%$[1]
$f(x) = cos(0.4 * sin(0.2 * x) + 3)$	$12.26\% \pm 2.12\%$	$13.23\% \pm 02.30\%$

[1]These high standard deviation values observed in the BP results indicate that the model produces unstable predictions, leading to significant fluctuations in performance. The corresponding outcome comparison plots (similar to the ones shown in Fig. 2).
[2]Observe MFAN is more stable and achieves lower error rates than BP on average for all these evaluations.

2.3 Online Continual Learning: Adapting to Dynamic Environments

This set of experiments focused on MFAN's performance in online continual learning (OCL) tasks, where the model was required to learn and adapt to a sequence of evolving trigonometric functions (Fig. 3). These tasks were designed to simulate dynamic environments where the data distribution changes over time, requiring the model to continuously update its knowledge and adapt to new, unseen data. We employ six tasks, each being a function defined in a domain of length 30π. These tasks are concatenated continuously to form a smooth task stream (TS). The TS is then divided evenly into multiple points, which are then grouped into data segments in a sliding window manner and each segment is a vector in $\mathbb{R}^{100}$. These data segments are fed into MFAN and BP models as input during the training and evaluation phases. The models each receive five consecutive data segments in the training phase and are then tested on the next upcoming segment in the evaluation phase. Again in a sliding window manner, the train and evaluate phases repeat until models reach the end of a TS. For detailed schematics, refer to Fig. 3.

We designed three sets of TS to evaluate MFAN's performance in OCL and assess the models' ability to handle task transitions, adapt to unseen data (TS1), process data with various patterns (TS2), and reorganize after interruptions (TS3).

2.3.1 Sinusoidal Function Series (TS1)

TS1 involved a sequence of sinusoidal functions with varying frequency and amplitude, testing the model's ability to adapt to smooth but non-linear changes. The prediction plots Fig. 4a-TS1 show that MFAN learns

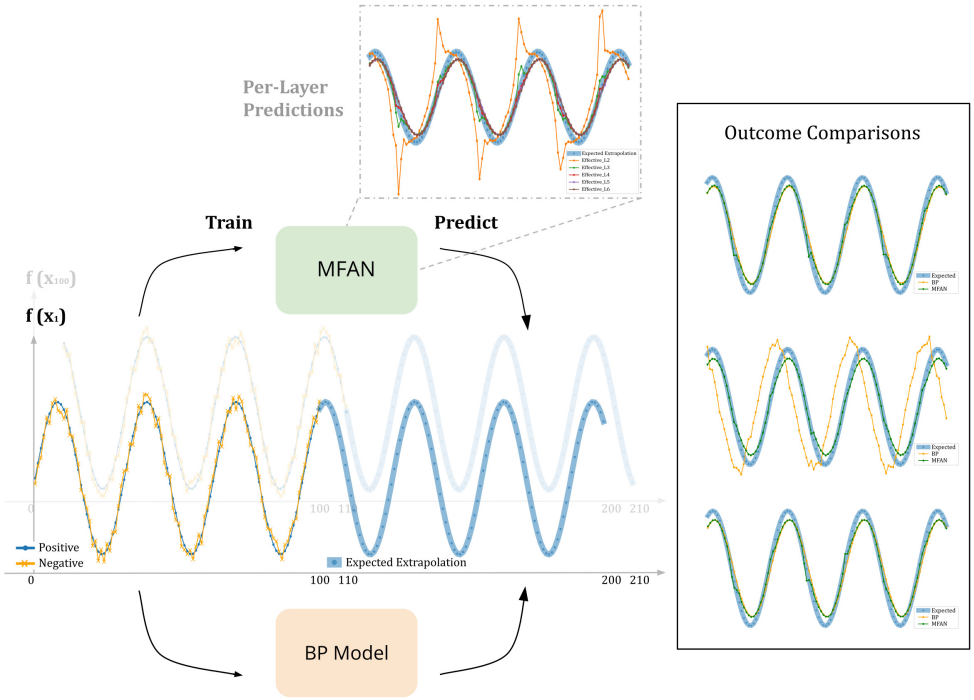

Figure 2: Schematic for function regression extrapolation task procedure. The positive data pairs, represented by the thin blue line, consist of $(x, y_{pos} = f(x) \in (\mathbb{R}^d, \mathbb{R}^d)$ where f is the target function and $d = 100$. The negative data pairs, represented by the thin orange line, consist of $(x, y_{neg} = y_{pos} + r \in (\mathbb{R}^d, \mathbb{R}^d)$ where r is a random offset for each data point and $d = 100$. The expected extrapolation data $(x_{test}, y_{test}) \in (\mathbb{R}^d, \mathbb{R}^d)$ in shown by the thick blue line. All datasets are constructed by utilizing a sliding window approach (e.g. training dataset x: $x_{i+1} = x_i + 0.1i$ with $x_0 = 0$ $i \in \mathbb{Z}, i \in [1, 100]$, training dataset $y_{pos} = f(x)$), with the training dataset in the domain $[0, 110]$, and the test dataset in domain $[100, 210]$. The first and the last set of data pairs for training and testing datasets are shown in the figure. After training, the performance of each layer of MFAN is computed and compared, giving guidance in choosing prediction at inference time. The training and evaluation processes are repeatedly performed to obtain average results, and model outcome comparisons are made. Some of the comparison plots are shown here as examples.

the first task slowly but handles task transitions well and adapts to new data quickly, producing stable and accurate predictions. BP learns the first task fast but struggles to adapt to new tasks.

The percentage error plots in Fig. 4b-TS1 show MFAN had frequent layer adjustment at the beginning of TS1. The per-layer error plots in Fig. 4c suggest that at the beginning, the shallow layer ($L1$) of MFAN had random predictions with very high errors. Deeper layers ($L2$, $L3$) are producing conservative predictions as their prior layer is not passing over meaningful directional information.

Then $L1$ learns discriminating features and makes better predictions than deeper layers which seems to overfit. Observe the effective layer selector of MFAN compares the layer performance during the latest training phase and select the best-performed layer to use during the evaluation phase, largely lowering the error rate. In contrast, sharp error rate increases and fluctuations are observed in BP, showing it failed adaptation.

2.3.2 Trigonometric Function Series (TS2)

TS2 introduced a wider variety of trigonometric functions. These functions featured more intricate patterns than TS1, requiring the model to capture more complex and diverse relationships between input and output variables (Fig. 4a-TS2).

As shown in Fig. 4b-TS2, MFAN maintained strong performance with a stable error rate as in TS1, showing it handles tasks with varied complexity or pattern steadily. In contrast, BP's error rates became increasingly unstable during transitions between tasks. When learning more complex data patterns like T3, it's clear from the error plot that BP had a rise in error compared to TS1.

2.3.3 Linear and Trigonometric Function Series (TS3)

TS3 inserted a linear function amid a series of trigonometric tasks to study models' reactions to sudden drastic changed data distribution (Fig. 4a-TS3). The simple linear function in the middle of TS3 disrupted the learning process, as the model had to adjust to a different type of relationship – linear – instead of periodic patterns of trigonometric functions, below referred to as a network "injury".

MFAN struggled with the linear task as it attempted to keep periodical predictions. Multiple layer adjustment attempts were observed during this period (Fig. 4b-TS3), showing MFAN is actively adjusting to recognize "injury". This also proves that MFAN's learning procedure has been hindered in this period. When switching back to trigonometric tasks, MFAN demonstrated fast adaptability and high accuracy. This shows that even though MFAN neuron mappings are stimulated during "injury", MFAN can reconnect and reorganize neurons quickly to recover from previous experiences and predict accurately again. Subsequent evaluations suggest MFAN regained stability as well. Such ability is also referred to as neural plasticity or brain plasticity Mateos-Aparicio & Rodríguez-Moreno (2019). In contrast, the BP model adapted quickly to the linear function and showed deteriorated performance after "injury". Its reliance on gradient-based optimization allowed it to quickly learn the linear relationship, reducing error rates swiftly. However, BP did not show neural plasticity and had an increase in error when shifting back to non-linear tasks.

These experiments highlight MFAN's potential for use in dynamic, real-world applications where the ability to adapt to evolving tasks is critical. MFAN's layered learning approach allows it to selectively focus on different aspects of the data, maintaining stability and performance when facing evolving tasks. MFAN also demonstrated neural plasticity to some extent, which is worth further investigation and may support its biological plausibility.

3 Discussion

MFAN presents advancements in neural network training, particularly for applications requiring continual learning and adaptability in dynamic environments. Through integrating contrastive and a multi-layer learning algorithm with dynamic layer adjustment, MFAN offers a promising alternative to traditional BP, demonstrating good stability, performance, and flexibility in handling both discrete and continuous data of given tasks.

MFAN excels in function regression, where it consistently outperformed BP by capturing complex, non-linear relationships and generalizing effectively to unseen data. The contrastive learning properties Chen et al. (2020); Du et al. (2024) at the core of MFAN could be one of the primary reasons behind generalizability, while the layer adjustment and independent layer learning characteristics inherited from FF also contribute to better performance.

MFAN's performance in OCL tasks further highlights its adaptability and stability. In the long-term with evolving data, MFAN generated more accurate and stable predictions compared to BP. Moreover, MFAN effectively adapts to varying task complexity and exhibits signs of neural plasticity — the ability of the nervous system to alter its activity in response to stimuli by reorganizing its structure, functions, or connections Mateos-Aparicio & Rodríguez-Moreno (2019). This resemblance to brain functionality could support the biological plausibility of MFAN with continued exploration.

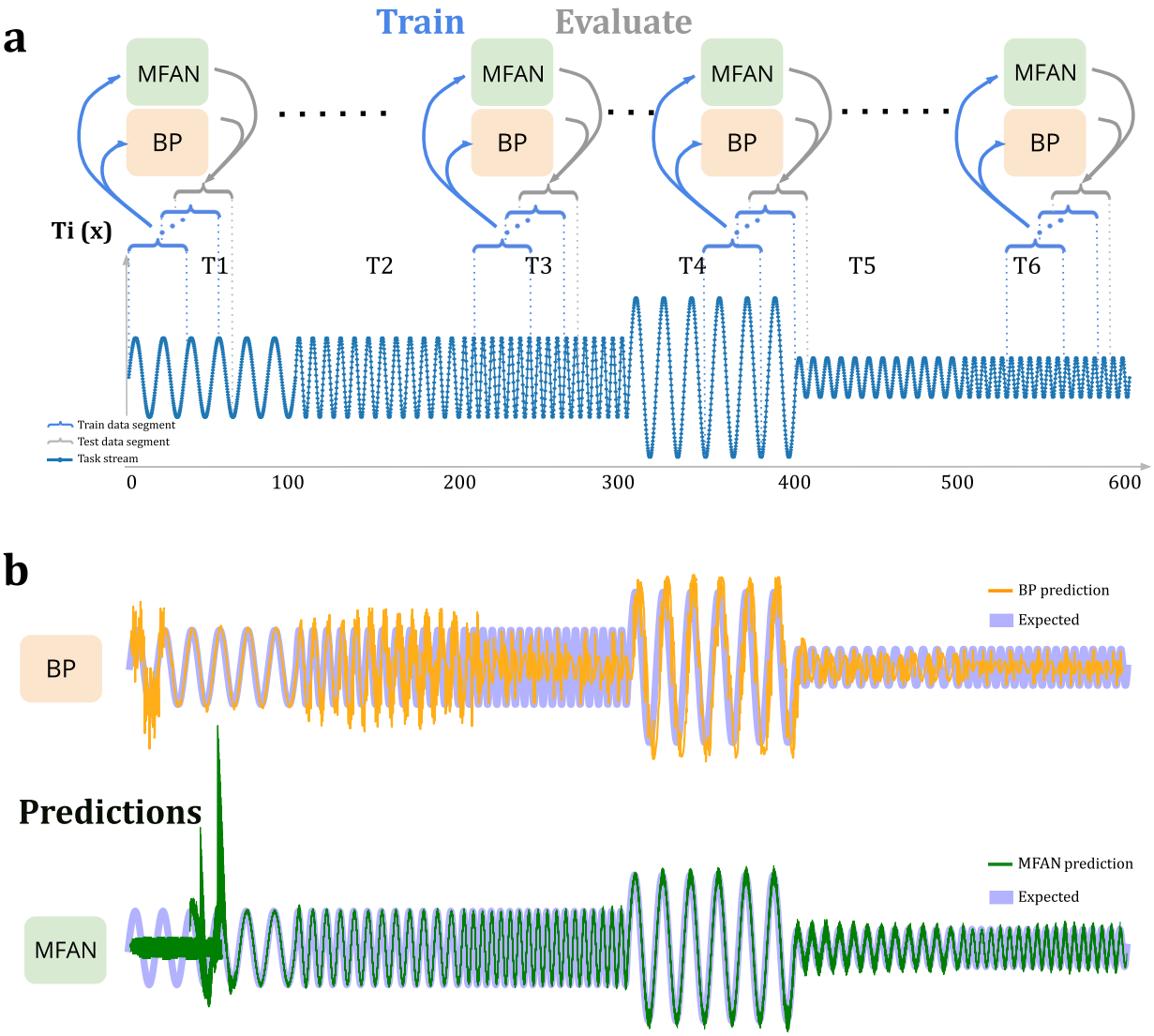

Figure 3: **a)** The Online Continual Learning (OCL) evaluation schematics. **a)** The OCL training and evaluation process. Six sinusoid task variations (Ti), each spanning a domain of length 30π, are concatenated continuously to form a smooth task stream. Evenly divide the task stream into 3000 points along the x-axis, with $p_i = (p_{x_i}, p_{y_i})$. The data segments are constructed by utilizing a sliding window approach as follows: $data_1 = [p_1, ..., p_{100}]$, $data_2 = [p_2, ..., p_{101}]$, keep shifting the window until construct $data_{2901} = [p_{2900}, ..., p_{3000}]$. The OCL train and evaluate process is structured such that the model is trained on five consecutive data segments, and subsequently, it predicts the next data segment in the sequence. For instance, the model is trained on $segment_1 to segment_5$, and then tasked with predicting $segment_6$. This procedure is repeated iteratively until all points in the data stream have been processed. **b)** Plots of predictions. Each model's predictions during the evaluation phases are collected and plotted against the expected value. Put plots for BP and MFAN together to make comparisons.

3.1 Optimization and Computation: Balancing Complexity and Efficiency

MFAN's flexibility in learning rate adjustment and layer-wise optimization allows independent tuning of components – $\boldsymbol{f_x}$, $\boldsymbol{f_y}$, and $\boldsymbol{f_y^{-1}}$ – making it responsive to varying data complexities. This adaptability enhances its performance on diverse tasks, though introduces additional hyper-parameters that require tuning.

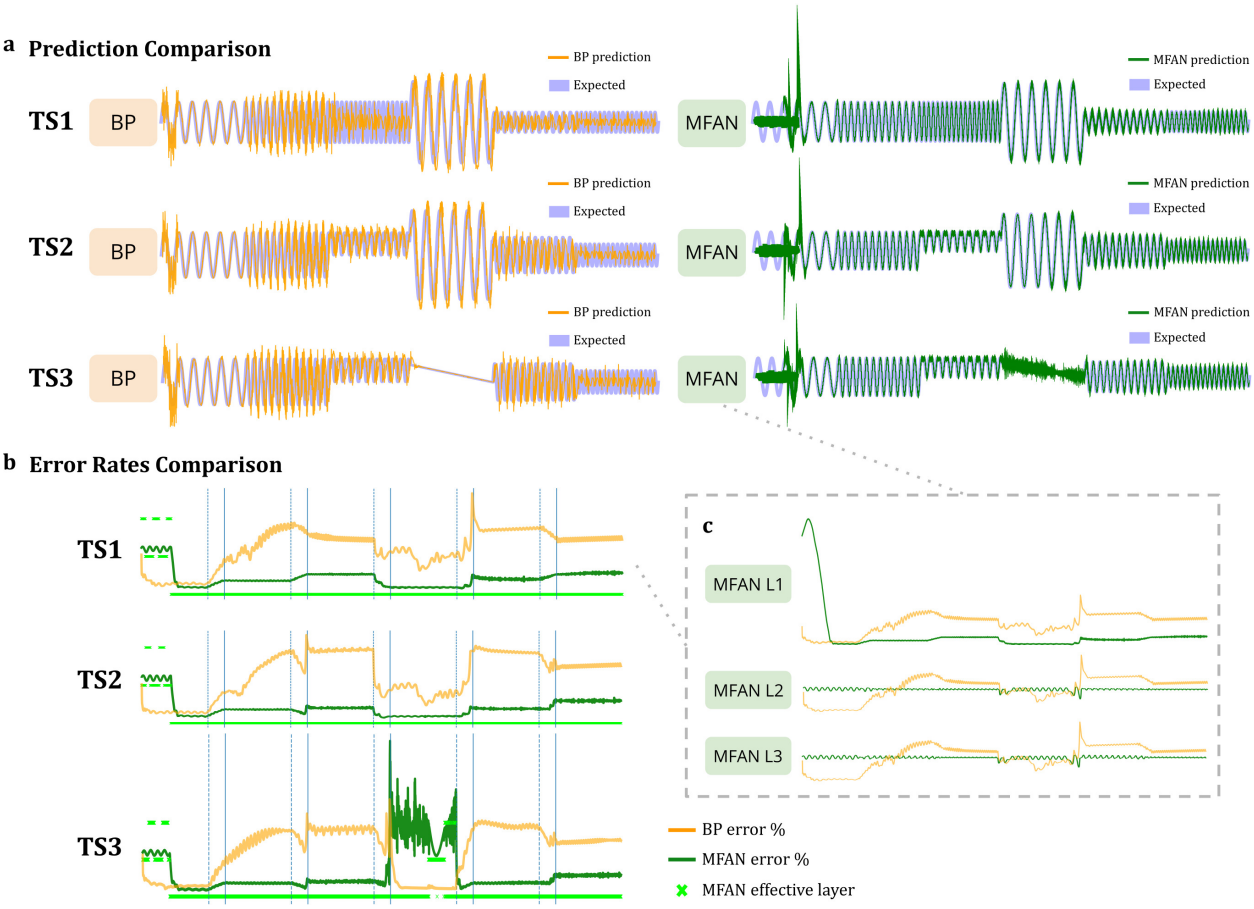

Figure 4: The Online Continual Learning (OCL) results. **a)** BP and MFAN predictions for all three task streams (TS) plotted against expected values. TS1 consists of sinusoid function variations, test models' ability to handle task transitions, and their adaptability to unseen data. MFAN shows stability and accurate long-term predictions, yet BP generates noisy predictions as tasks evolve. TS2 includes a broader range of trigonometric functions, checks models' ability to deal with data with various patterns, and increases the complexity of underlying functional relationships. MFAN steadily handles complexity change, whereas BP prediction becomes more noisy. TS3 introduces a linear function amid periodical trigonometric functions as a form of injury. It tests model behaviors during and post-injury. MFAN demonstrates neural plasticity to some extent as it's able to recover from injury through what seems to be a reorganization or remapping of neuron connections. In contrast, BP demonstrated limited adaptability, exhibiting deteriorated performance and inability to recover effectively post-injury. **b)** The corresponding percentage error rates for evaluations in **(a)**, cross-proving the prediction plot performances with detailed numerical comparisons. **c)** The per-layer error rate of MFAN compared with BP for TS1. Observed that MFAN is actively selecting layers to stabilize error.

While MFAN training demands more computational resources than BP due to its complex architecture, this investment is justified by its substantial advantages in adaptability and predictive accuracy in dynamic settings.

3.2 Biological Plausability

We would like to discuss three main aspects of biological plausibility Lv et al. (2024): (i) asymmetry of weights; (ii) local error representation; and (iii) non-parallel training.

- **Asymmetry of weights:** Mostly, the artificial neurons in the forward path transfer their synaptic weights back along a feedback path, a mechanism called weight transport, which is integral to back-propagation. This practice enables error correction by adjusting weights based on feedback; however, it is considered biologically unrealistic. In real neural systems, neurons are unlikely to share exact synaptic weights in such a precise manner by making direct weight transport an implausible model of natural learning processes. The MFAN model addresses this limitation by implementing localized feedback mechanisms. Rather than relying on extensive feedback paths, MFAN facilitates per-layer feedback and updates. This approach allows each layer to make targeted adjustments in response to its own performance, reflecting a more biologically plausible learning process. By aligning with how neurons may adapt their connections based on immediate local information, MFAN enhances efficiency and improves the model's alignment with natural learning systems.
- **Local error representation:** Biological synapses are thought to adjust their strength using only local information, without relying on a global error signal. This differs from the gradient descent method in artificial neural networks, where the direction of the error gradient is typically calculated based on global information. With the MFAN, error representations are localized. Each layer can adjust its weights based on feedback from its immediate inputs and outputs, rather than depending on a comprehensive network-wide error signal. This localized approach mirrors the adaptive processes of real neurons, thereby making the MFAN model more robust and biologically plausible.
- **Non-parallel training:** Traditional training methods typically necessitate a clear separation between forward and backward propagation phases, a structure absent in biological learning processes. In contrast, bio-plausible approaches are being investigated for their potential to streamline learning into more continuous, potentially overlapping phases that more closely resemble the dynamics of biological learning. The MFAN model excels in OCL tasks (subsection 2.3), achieving significant improvements over traditional back-propagation methods. Its architecture supports real-time adaptation, allowing the model to update weights as new data streams come, rather than waiting for batch processing. Moreover, MFAN incorporates contrastive learning to enhance the model's ability to differentiate relevant features from irrelevant ones. This mechanism not only aids in real-time adjustments but also allows the model to strengthen important connections while weakening those that are less useful. By reinforcing valuable representations and adapting to new data, MFAN effectively manages the balance between retaining previously acquired knowledge and accommodating new information.

MFAN approach lacks some other biologically plausible features like the use of spikes and unsigned error signals. Spiking neurons enable temporal dynamics and energy efficiency in natural learning processes, while unsigned error signals offer a direction-neutral approach to synaptic adjustments. Fortunately, MFAN's layer-wise-independent design provides flexibility for re-architecture. Each layer acts as a self-contained unit with its own input and feedback loops, making it adaptable to integrating additional components and performing localized adjustments – such as incorporating spiking neural networks Subbulakshmi Radhakrishnan et al. (2021); Yin et al. (2023), or configuring specific layers for magnitude-only gradient updates – to enhance biological realism.

4 Conclusion

In this work, we introduced the Metamorphic Forward Adaptation Network (MFAN), an innovative neural architecture designed to overcome the limitations of traditional back-propagation and Forward-Forward algorithms. MFAN demonstrates exceptional flexibility and robustness in tasks requiring continual learning and adaptation, particularly through its dynamic layer depth adjustment and contrastive learning mechanisms. It outperforms baseline models in function regression and online continual learning by delivering stable and accurate predictions even in complex, evolving environments.

Beyond its technical advancements, MFAN represents a step toward more biologically plausible neural networks. Unlike conventional models relying on global error signals, MFAN employs a localized, layer-wise learning approach akin to how biological systems may process and adapt to information. This allows MFAN

to dynamically adjust to changing data streams aiming to reflect biological learning to reorganize its neural connections in response to new experiences – a phenomenon known as neural plasticity.

Looking ahead, future research could further explore the integration of additional biologically inspired features, such as spiking neurons or more advanced forms of neural plasticity, to further align MFAN with the adaptive learning processes observed in natural systems.

References

Md Atik Ahamed, Jin Chen, and Abdullah-Al-Zubaer Imran. Forward-forward contrastive learning. *arXiv preprint arXiv:2305.02927*, 2023. doi: 10.48550/arXiv.2305.02927.

Andri Ashfahani and Mahardhika Pratama. Autonomous deep learning: Continual learning approach for dynamic environments. In *Proceedings of the 2019 SIAM international conference on data mining*, pp. 666–674. SIAM, 2019. doi: 10.1137/1.9781611975673.7.

Yoshua Bengio, Dong-Hyun Lee, Jorg Bornschein, Thomas Mesnard, and Zhouhan Lin. Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*, 2015. doi: 10.48550/arXiv.1502.04156.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020. URL `https://arxiv.org/abs/2002.05709`.

Shibhansh Dohare, J Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026):768–774, 2024. doi: 10.1038/s41586-024-07711-7.

Jun Du, Jianhang Jin, Jian Zhuang, and Cheng Zhang. Hierarchical graph contrastive learning of local and global presentation for multimodal sentiment analysis. *Scientific Reports*, 14(1):5335, 2024. doi: 10.1038/s41598-024-54872-6.

Saumya Gandhi, Ritu Gala, Jonah Kornberg, and Advaith Sridhar. Extending the forward forward algorithm. *arXiv preprint arXiv:2307.04205*, 2023. doi: 10.48550/arXiv.2307.04205.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 297–304. JMLR Workshop and Conference Proceedings, 2010.

Guy Hay and Nir Sharon. Function extrapolation with neural networks and its application for manifolds. *arXiv preprint arXiv:2405.10563*, 2024. doi: 10.48550/arXiv.2405.10563.

Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022. doi: 10.48550/arXiv.2212.13345.

Geoffrey E Hinton, Terrence J Sejnowski, et al. Learning and relearning in boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1(282-317):2, 1986. doi: 10.7551/mitpress/3349.003.0005.

Stefano Izzo, Fabio Giampaolo, Diletta Chiaro, and Francesco Piccialli. Shelob-ffl: addressing systems heterogeneity with locally backpropagated forward-forward learning. In *2024 IEEE 44th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pp. 23–28. IEEE, 2024. doi: 10.1109/ICDCSW63686.2024.00010.

Ali Karimi, Ahmad Kalhor, and Melika Sadeghi Tabrizi. Forward layer-wise learning of convolutional neural networks through separation index maximizing. *Scientific Reports*, 14(1):8576, 2024. doi: 10.1038/s41598-024-59176-3.

Deepak Kumar, Umang Goswami, Hariprasad Kodamana, Manojkumar Ramteke, and Prakash Kumar Tamboli. Variance-capturing forward-forward autoencoder (vffae): A forward learning neural network for fault detection and isolation of process data. *Process Safety and Environmental Protection*, 178:176–194, 2023. doi: 10.1016/j.psep.2023.07.083.

Alfirna Rizqi Lahitani, Adhistya Erna Permanasari, and Noor Akhmad Setiawan. Cosine similarity to determine similarity measure: Study case in online essay assessment. In *2016 4th International conference on cyber and IT service management*, pp. 1–6. IEEE, 2016. doi: 10.1109/CITSM.2016.7577578.

Changze Lv, Yufei Gu, Zhengkang Guo, Zhibo Xu, Yixin Wu, Feiran Zhang, Tianyuan Shi, Zhenghua Wang, Ruicheng Yin, Yu Shang, et al. Towards biologically plausible computing: A comprehensive comparison. *arXiv preprint arXiv:2406.16062*, 2024. doi: 10.48550/arXiv.2406.16062.

Pedro Mateos-Aparicio and Antonio Rodríguez-Moreno. The impact of studying brain plasticity. *Frontiers in cellular neuroscience*, 13:66, 2019. doi: 10.3389/fncel.2019.00066.

Abel A Reyes-Angulo and Sidike Paheding. Forward-forward algorithm for hyperspectral image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3153–3161, 2024.

Shiva Subbulakshmi Radhakrishnan, Amritanand Sebastian, Aaryan Oberoi, Sarbashis Das, and Saptarshi Das. A biomimetic neural encoder for spiking neural network. *Nature communications*, 12(1):2143, 2021. doi: 10.1038/s41467-021-22332-8.

Yanshuo Wang, Ali Cheraghian, Zeeshan Hayder, Jie Hong, Sameera Ramasinghe, Shafin Rahman, David Ahmedt-Aristizabal, Xuesong Li, Lars Petersson, and Mehrtash Harandi. Backpropagation-free network for 3d test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23231–23241, 2024.

Keyulu Xu, Mozhi Zhang, Jingling Li, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: from feedforward to graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2021. doi: 10.48550/arXiv.2009.11848.

Bojian Yin, Federico Corradi, and Sander M Bohté. Accurate online training of dynamical spiking neural networks through forward propagation through time. *Nature Machine Intelligence*, 5(5):518–527, 2023. doi: 10.1038/s42256-023-00650-4.

A Appendix

A.1 MNIST Classification

Models were trained on 60,000 MNIST images and tested on 10,000 images. Model configurations are outlined in Table A.1 below. Accuracy and latent space distribution were analyzed.

MNIST Model Setup		
	MFAN	**BP Model**
Network depth	3-layer	2-layer
Network width	$[500, 300, 300]$	$[500, 300]$
Learning rate	$5e^{-5}$, $5e^{-4}$, $5e^{-2}$	$5e^{-4}$
$[a_1, a_2]$	$[0.6, 0.6]$	/
Number of train data	60000	60000
Number of test data	10000	10000
Encoder loss	Cosine Similarity	/
Decoder loss	Cross Entropy	Cross Entropy

A.2 Function Regression

Models were trained and tested on self-generated function regression datasets. Models were assessed on generalizability, the understanding of underlying relationships, and the ability to handle functions with various complexity. Model setups are listed in Table A.2.

FR Model Setup		
	MFAN	**BP Model**
Network depth	6-layer	3-layer
Network width	$[128, 128, 64, 64, 32, 32]$	$[128, 64, 32]$
Learning rate	$1e^{-5}$, $1e^{-4}$, $1e^{-2}$	$1e^{-5}$
$[a_1, a_2]$	$[0.6, 1]$	/
Number of data	100	100
Length of data	100	100
x **examples:** x_1**,** x_2	$[0, 1, ..., 100]$, $[0.1, 1.1, ..., 100.1]$	$[0, 1, ..., 100]$, $[0.1, 1.1, ..., 100.1]$
Train x **domain**	$[0, 110]$	$[0, 110]$
Test x **domain**	$[100, 210]$	$[100, 210]$
Range y_{neg}	$[y_{pos_{min}} - 0.1, y_{pos_{min}} - 0.03]$ $\cup [y_{pos_{min}} + 0.03, y_{pos_{max}} + 0.1]$	/
Encoder loss	Cosine Similarity	/
Decoder loss	Root Mean Square	Root Mean Square

A.3 Online Continual Learning

Models were trained and tested on self-generated function regression datasets. Models were assessed on generalizability, the understanding of underlying relationships, and the ability to handle functions with various complexity. The model setup is listed in Table A.2. Models were evaluated by performing continual learning of three task streams in dynamic environments. The models' predictions and error rates were monitored. Models' long-term dynamic adaptability, stability, and neural plasticity were evaluated. Detailed model configurations are shown in Table A.3.

Model Setup		
	MFAN	**BP Model**
Network depth	3-layer	3-layer
Network width	$[64, 32, 16]$	$[64, 32, 16]$
Learning rate	$1e^{-5}$, $1e^{-5}$, $1e^{-4}$	$1e^{-1}$
$[a_1, a_2]$	$[0.6, 1]$	/
Number of data	2901	2901
Length of data	100	100
x **examples:** x_1**,** x_2	$[0, 0.2, ..., 18.7]$, $[0.2, 0.4, ..., 18.9]$	$[0, 0.2, ..., 18.7]$, $[0.2, 0.4, ..., 18.9]$
Range y_{neg}	$[y_{pos_{min}} - 0.1, y_{pos_{min}} - 0.03]$ $\cup [y_{pos_{min}} + 0.03, y_{pos_{max}} + 0.1]$	/
Encoder loss	Cosine Similarity	/
Decoder loss	Root Mean Square	Root Mean Square