ARXSA: A General Negative Feedback Control Theory in Vision-Language Models

Anonymous ACL submission

Abstract

The Transformer model has been increasingly applied across various domains, driven by the self-attention mechanism, which offers robust data processing capabilities and has substantially contributed to the advancement of the model. In the self-attention mechanism, three core matrices from the same data batch are computed together to determine correlations between input elements. Drawing inspiration from the efficiency and stability conferred by negative feedback structures in predictive control systems, the concept of vertical training was introduced to integrate data from multiple batches. Accordingly, this paper proposes an autoregressive with exogenous inputs (ARX) approach for the self-attention mechanism, transforming the Encoder block into a negative feedback predictive control system. A network architecture based on this method is also proposed, enabling the autoregressive with exogenous inputs for self-attention to transmit data from batches at previous time points. The effectiveness of the proposed approach is validated through comparative experimental evaluations.

1 Introduction

In recent years, neural network research had gradually become central to advancements in artificial intelligence(Voulodimos et al., 2018; Zhao et al., 2024; González-Sabbagh and Robles-Kelly, 2023). Computer Vision (CV) and Natural Language Processing (NLP) had also assumed significant roles in daily life(Fanni et al., 2023; Khurana et al., 2023). Among the developments in these fields, the emergence of self-attention (SA) mechanisms and Transformer networks had addressed the limitations imposed by short memory in traditional attention mechanisms(Vaswani et al., 2017). The application of SA to the CV domain had similarly enhanced network performance(Li et al., 2023; Sun et al., 2023). However, existing research on SA mechanisms primarily concentrated on improving computational efficiency and optimizing performance(Zaheer et al., 2020; Hassani et al., 2023). For instance, the Swin Transformer significantly reduced computational overhead by minimizing less relevant correlations in the SA mechanism, yielding improved performance on several benchmark datasets(Liu et al., 2021; Dong et al., 2022). These computational optimizations were generally predicated on the importance of pixel correlation calculations. While these methods reduced computational load for distant pixel relationships, they concurrently strengthen the correlation between neighboring pixels. Some approaches also lowered the computational complexity by reducing the dimensionality of low-rank core matrices, effectively adjusting the image size(Shen et al., 2021; Mayer et al., 2022). Although these techniques had undeniably optimized the SA mechanism, there remained a notable lack of a robust theoretical foundation for SA and Transformer models. As a result, the tuning of Transformer parameters and architectures continued to rely heavily on empirical experimentation.

To enhance the controllability of neural networks, several studies had explored the integration of neural networks with control theory, beginning with common residual blocks, in an effort to uncover underlying patterns(Chen et al., 2018; Zhang et al., 2021; Chien and Chen, 2021; Chen et al., 2024). Control theory, having evolved rapidly over the past decades from a branch of mathematics into a distinct and widely applied discipline, was capable of integrating principles from numerous fields. Predictive control theory, in particular, offered the advantage of improving both control performance and efficiency by forecasting future system behaviors and optimizing control inputs(Maciejowski and Huzmezan, 2007; Babayomi et al., 2023). In pursuit of imparting controllability to neural networks, this study examines the self-attention mechanism from the perspective of predictive control systems. It is observed that the modified SA mechanism can be conceptualized as an autoregressive with exogenous inputs (ARX) model(Huang et al., 2023).

To introduce predictability into certain neural networks, this paper re-evaluates the connections between the self-attention mechanism and various blocks within Transformers. By integrating the residual structure in the Encoder block with a modified SA mechanism, a method has been developed that enhances training accuracy with minimal additional computational cost. Specifically, the proposed method, termed autoregressive with exogenous inputs for self-attention (ARXSA), gains temporal characteristics by constructing the self-attention mechanism through core matrices generated at different time points. These matrices are combined with the current matrix through a recurrent process, imparting temporal characteristics to the exogenous inputs as well. To assess the feasibility of this approach, the autocorrelation function (ACF) is employed to validate its autoregressive properties(Podulka et al., 2023; Wu et al., 2023). The results confirm that the ARXSA method exhibits autocorrelation, aligning it with the ARX model. This suggests that ARXSA can adjust network outputs based on historical data. To distinguish between conventional training methods and those incorporating historical data, the concept of unit time in the training process is introduced, defined based on batches. Consequently, training methods are classified into horizontal and vertical training, depending on the unit time. By employing the vertical training concept, the ARXSA method transforms the Encoder block into a negative feedback predictive control system. The feedback path of ARXSA improves both the efficiency and robustness of the overall network(Wu et al., 2020; Wang et al., 2017, 2020). Furthermore, ARXSA is integrated into Transformers to enhance their operational efficiency. The resulting Transformer can be interpreted as a predictive control system with autoregressive properties. Extensive experiments on several widely-used benchmark datasets demonstrate the efficacy of the proposed Transformer in image classification and object detection tasks. Additionally, its generality is validated in networks similar to Swin Transformer. The contributions are summarized as follows:

- The concept of vertical training is introduced, differentiating it from traditional training methodologies. This approach enables the concurrent computation of data across multiple batches, significantly improving both the robustness and efficiency of the network.

- A novel theory is proposed to transform the self-attention mechanism into an autoregressive (AR) model, unveiling the strong connection between the SA mechanism and predictive control systems. This theory not only provides a robust explanation for the performance improvements achieved by the ARXSA mechanism in Transformers but also establishes a theoretical foundation for the SA mechanism.

- Leveraging the insights gained from the ARXSA perspective, a Transformer backbone network grounded in predictive control theory is developed. This backbone network, named ARXFormer, integrates principles from both disciplines, offering a theoretically driven and highly effective approach.

2 Method

This section will systematically introduce predictive control theory, the ARX model, and the ACF, detailing the relationship between predictive control and the ARX model through illustrative examples. Next, the concept of unit time, which is defined in relation to batch processing in neural network training, will be introduced. The unit time will be followed by a description of two distinct training methods based on the unit time concept. Subsequently, the integration of the ARX model with the self-attention mechanism to form the ARXSA method will be explained, and its autocorrelation properties will be validated by ACF. Lastly, a comprehensive overview of the network architecture incorporating the ARXSA method into the Transformer will be provided.

2.1 Control Systems and ARX Model

In practical applications, control systems exhibit considerable diversity, with varying methods employed across different systems(Gong et al., 2020; Huang et al., 2024). Furthermore, control problems in real-world scenarios are often continuous, as systems require ongoing feedback from their outputs(Esterhuizen et al., 2020; Zhang et al., 2022; Tang et al., 2018). However, solving continuous problems is highly complex. It not only requires extensive computations but also involves cases where no solutions exist at extremum points. To mitigate the impact of solving continuous functions on algorithms and neural networks, we analyze the problem from a discrete perspective, leveraging the independence of batches. As a kind of statistical model, the autoregressive model is a prediction method based on time series(Dijk et al., 2002). The basic form of the model is as follows:

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + ... + a_n y_{t-n} + e_t, \quad (1)$$

where $y_{t-i}(i = 1, 2, ..., n.)$ is the output of the system at time $t - i$ $a_i(i = 1, 2, ..., n.)$ is the coefficient of the output and e_t is the error term. Therefore, the output of the AR model at time t is the weighted sum of the outputs of multiple historical moments.

The ARX model is one of the predictive control methods, combining the AR part and the exogenous input (X) part. Its model can be represented as:

$$\begin{aligned} y_t &= a_1 y_{t-1} + a_2 y_{t-2} + ... + a_n y_{t-n} \\ &\quad + b_1 u_{t-1} + b_2 u_{t-2} + ... + b_m u_{t-m} \\ &\quad + e_t, \end{aligned} \quad (2)$$

where $y_{t-i}(i = 1, 2, ..., n-1)$ is the output of the system at time $t - i$, $u_{t-j}(j = 1, 2, ..., m-1)$ is the exogenous input of the system at time t, and e_t is the error term. Meanwhile, i and j are the autoregressive order (AR order) and the exogenous variable order (X order) of the model. Based on AR model, ARX model adds exogenous input which also has autoregressiveness.

2.2 Autocorrelation Function

The autocorrelation function is applied to determine whether there is correlation at different time lags in a time-dependent sequence. Specifically, the ACF assesses whether there is autocorrelation by inputting the historical data of a variable itself. The basic form of the ACF is as follows:

$$\rho_h = \frac{Cov(X_t, X_{t-h})}{\sqrt{Var(X_t)Var(X_{t-h})}}, \quad (3)$$

where $X_i(i = 1, 2, ..., n)$represents the time series being evaluated, and h represents the lag order. Since the epochs of neural networks are often numerous, the sample autocorrelation function can also be employed. The expression is as follows:

$$\rho_h = \frac{\mathbf{E}[(X_{t+h} - \mu)(X_t - \mu)]}{\mathbf{E}[(X_t - \mu)^2]}, \quad (4)$$

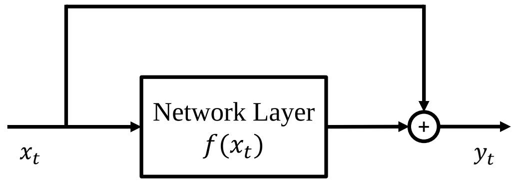

Figure 1: Residual structure of neural network.

where μ is the sample mean and $\mathbf{E}[]$ is the mathematical expectation of the time series.

The purpose of the ACF is to determine whether a time series exhibits autocorrelation. The necessary and sufficient conditions for employing ACF on a time series are as follows:

- The mean and variance of the time series must be constant.
- There should be no missing values in the time series data.
- The length of the time series must be sufficiently long.

The calculated results range is $(-1 \leq \rho_h \leq 1)$. Meanwhile, there is a positive correlation when $(0 < \rho_h < 1)$, there is a perfect positive correlationwhen $\rho_h = 1$, there is a negative correlation when $(-1 < \rho_h < 0)$, there is a perfect negative correlation when $\rho_h = -1$. and it indicates that there is no autocorrelation when $\rho_h = 0$.

2.3 Horizontal Training and Vertical Training

Currently, the residual structure has been adopted by most neural networks(He et al., 2016). Recent work has demonstrated that residual structure can be regarded as a special dynamic system(Weinan, 2017; Meunier et al., 2022; Zhu et al., 2023). For a basic residual block, it can be written as follows during a batch of training:

$$y_t = x_t + f(x_t), \quad (5)$$

where x_t is the input of the residual structure, $f(x_t)$ is the method of parameter training in the residual structure, and y_t is the result of the sum of the parameters after training and the input matrix of the same shape.

The residual block in Fig 1 is the SA mechanism in the Encoder block(Dosovitskiy et al., 2020; Liu et al., 2021). In addition to the multi-head self-attention (MHSA) layer, this block also includes a layer normalization (LN) layer and a Dropout layer.

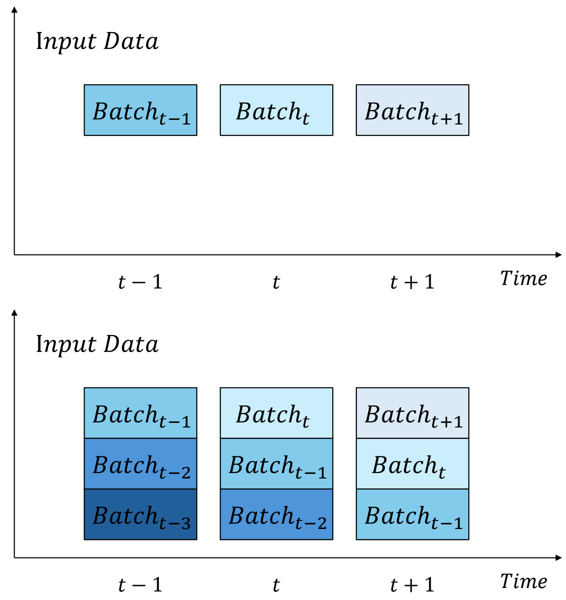

Figure 2: Comparison between horizontal training and vertical training based on the concept of unit time.

The LN is a normalization method that is independent of the input content, and its computations on the matrix are limited to scaling up or down proportionally. Additionally, the output matrix of the SA mechanism represents the significance of inter-pixel relationships, which is not affected by proportional scaling. Dropout is a method applied to prevent overfitting by effectively multiplying a randomly selected subset of parameters by zero. The randomness in parameter computation serves as a measure of the networks robustness. Within a controlled range, the randomness can be regarded as an error that follows a Gaussian distribution. Therefore, both the LN module and the dropout module do not affect the computation of the residual block. For the residual structure in Fig 1, it can be written similarly to equation (5) as follows:

$$y_t = x_t + f_{SA}(x_t), \tag{6}$$

where $f_{SA}(x_t)$ compose LN, MHSA and Dropout layers.

The residual structure effectively prevents accuracy from decreasing when the network depth is expanded(He et al., 2016). However, the residual structures ability to adjust and trace network parameters exists only within the computation process of a single batch. The time required for training each batch in the network is not always the same, so that real time cannot be defined as the unit of time to segment the training progress. Fortunately, except for the last batch, the size of each batch remains constant throughout the training process. Therefore, the period from when first batch enters the network to when the second batch enters the network was defined as a unit of time, denoted as t.

In a typical network, input data is divided into batches of equal size during the training process. These batches form a sequence that waits and enters the network sequentially for computation. The distribution of the input data sequence during the training process is shown in the upper figure of Fig. 2. In the figure, the horizontal axis represents the unit time, while the vertical axis represents all the input data. Since the unit of time is defined based on batches, the distribution of adjacent batches during training appears as a linear function with a near-zero constant term. However, batches are independent of each other, and at the given time t, only $Batch_t$ is processed by the network. We define the direction of the sequence of independent batches over time as the horizontal direction. The training within one epoch is referred to as horizontal training (HT).

Conversely, as shown in the lower figure of Fig 2, when information from multiple batches is input into the network simultaneously during training, the batch sequences are combined. In the figure, the horizontal axis still represents the unit time, and the vertical axis represents all the input data. The number of batches included at the same time is referred to as the order. The below figure shows the distribution of third-order input data at times $t-1$, t, and $t+1$. At time t, the information input into the network consists of $Batch_t$, $Batch_{t-1}$ and $Batch_{t-2}$. At the sequence, the input data at any given moment is distributed vertically. Therefore, training within one epoch in this manner is referred to as vertical training (VT).

2.4 Self-attention with ARX Model

In this section, the SA mechanism is employed to integrate current and historical data, allowing the networks training method to align with VT. Then, when the AR order is set to 1, it will be demonstrated that the Encoder block, employing the ARXSA method, functions as a negative feedback predictive control system. Consequently, the residual structure of the Encoder block is transformed into an ARX model.

The self-attention mechanism can be represented as:

$$SA = softmax(\frac{QK^T}{\sqrt{d_k}})V, \tag{7}$$

where the three matrices Q, K, and V are generated by the input data of the same batch(Vaswani et al., 2017).

Algorithm 1 The process of insert K_{t-1} matrix

Generate Q_t, K_t, V_t from $Batch_t$
if $t! = 1$ **then**
  $L_t = \alpha K_t + \beta K_{t-1}$
  $K_{t-1} = L_t$
else
  $L_t = \alpha K_t + \beta K_t$
  $K_{t-1} = L_t$
end if
$SA = softmax(\frac{Q_t L_t^T}{\sqrt{d_k}})V_t$

The overall process of ARXSA is shown in Algorithm 1. Specially, when $t = 1$, the K_{t-1} matrix does not participate in the computation. Meanwhile, the matrix K_{t-1} does not participate in backpropagation during the training process, but is only responsible for transferring the matrix K at time $t-1$. Let the matrix obtained by adding K_t and K_{t-1} be L_t, then L_t is expressed as:

$$L_t = \alpha K_t + \beta K_{t-1} + e_t, \tag{8}$$

where e_t is the independent and identically noise in training process(Semenova et al., 2022; Kosson et al., 2024). α and β are the coefficients of K_t and K_{t-1}. From equation (8), it can be seen that with the addition of the K_{t-1} matrix, the L_t matrix is obtained by weighting matrices containing information from both time t and $t-1$. According to Section 2.3, the ARXSA method forms a VT setup in Fig 2 where information from two different time points appears simultaneously within the network.

Based on the associative property of matrix multiplication, Q_t can be added after separately multiplying with K_t and K_{t-1}. That process can be written as:

$$Q_t L_t^T = \alpha Q_t K_t^T + \beta Q_t K_{t-1}^T + e_t, \tag{9}$$

where e_t is the independent and identically noise. Due to the tiny value of e_t in equation (8), multiplying it with Q_t does not significantly affect the overall system. Furthermore, Q_t, K_t and K_{t-1} are independent of each other. $Q_t K_t$ can be considered as the autoregressive term in the ARX model, while $Q_t K_{t-1}$ can be considered as the exogenous input term.

Therefore, the training process of the Encoder block according to Fig 1 is shown in Fig 3. The figure illustrates the data transmission direction at time t for an lag order of 1. Once the L_{t-1} matrix is generated at time $t-1$, it is directly passed to the data flow at time t without any additional computation or backpropagation and is processed together in the Encoder block. The L_{t-1} matrix serves as a feedback path that returns information to the forward path entering the Encoder block. Meanwhile, the upper and lower parts represent the same Encoder Block. The "Encoder Block x" in the figure can be any Encoder Block in the network, because every Encoder Block performs the same processing. The L_{t-1} obtained after the input data passes through the MHSA will be passed to the input part of the MHSA at the next moment. According to equation (2) and (6), the process of Fig 3 can be expressed as:

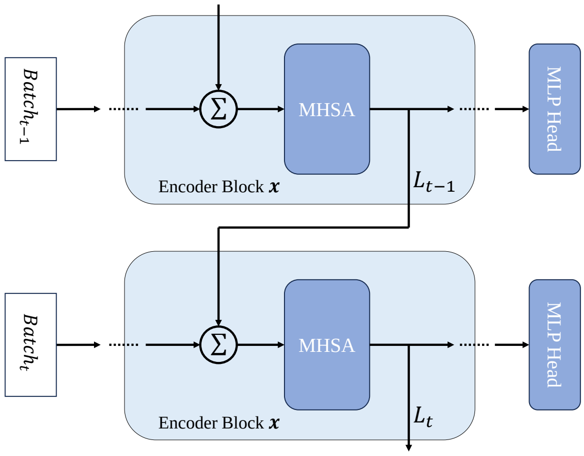

Figure 3: Negative feedback structure of Encoder block based on ARXSA.

$$\begin{aligned} y_t &= x_t + f_{ARXSA}(x_t) \\ &= x_t + Dropout(ARXSA(Linear(LN(x_t)))) \\ &= x_t + ARXSA(Linear(x_t)) \\ &= x_t + ARXSA(Q_t, K_t, K_{t-1}, V_t) \\ &= x_t + softmax(\frac{Q_t L_t^T}{\sqrt{d_K}})V_t, \end{aligned} \tag{10}$$

where the $Linear()$ function maps x_t into Q_t, K_t and V_t through the parameters. Therefore, the assumption that the Encoder block with the ARXSA method aligns to the negative feedback predictive control system has been validated.

Additionally, to prove that the ARXSA method belongs to the ARX model category, the following additional conditions must be verified:

- The input sequence must be stationary.

- The model needs a definite AR order.
- The error terms must be independent.
- The input sequence must exhibit autocorrelation.

Firstly, the Q_t, K_t and V_t matrices generated by the input data through the fully connected layer conform to the Kaiming normal distribution(He et al., 2015), thus ensuring that the matrices involved in the ARXSA method are stationary. Secondly, both the AR order and lag order of the ARXSA method are set to 1 in this work. Thirdly, most errors generated during the computation of neural networks are independently and identically distributed Gaussian noise, whose impact on the network is within an acceptable range(Ghosh et al., 2017; Seltzer et al., 2013). Finally, to determine whether the ARXSA method meets the conditions of the ARX model, it is necessary to employ the ACF to prove the autocorrelation of the K_t matrix.

Based on (4), the ACF of ARXSA can be written as:

$$\rho_h = \frac{\mathbf{E}[(L_{t+h} - \mu)(L_t - \mu)]}{\mathbf{E}[(L_t - \mu)^2]}, \tag{11}$$

where h is the lag order of the ARXSA, μ is the mean of the input data and the $textbfE[]$ function represented the function for finding the mathematical expectation. With (8), the expected value of the numerator in (11) can be expanded as:

$$\begin{aligned}\mathbf{E}&[(L_{t+h} - \mu)(L_t - \mu)] \\ &= \mathbf{E}[(\alpha K_{t+h} + L_{t+h-1} - \mu)] \\ &\quad * \mathbf{E}[(L_t + \beta K_{t-1} - \mu)],\end{aligned} \tag{12}$$

where α and β are the coefficients of K_t and K_{t-1}. Since the objects in each category of images in the database are similar, K_{t+h} is approximately equal to K_t. Moreover, K_t can be approximated as the mean vector $\overline{K}$. The (12) can be divided as:

$$\begin{aligned}\mathbf{E}&[(L_{t+h} - \mu)(L_t - \mu)] \\ &\approx \mathbf{E}[(\alpha\overline{K} - \mu)(\alpha\overline{K} - \mu)] \\ &+ \mathbf{E}[(\alpha\overline{K} - \mu)(\beta K_{t-1} - \mu)] \\ &+ \mathbf{E}[(L_{t+h-1} - \mu)(\alpha\overline{K} - \mu)] \\ &+ \mathbf{E}[(L_{t+h-1} - \mu)(\beta K_{t-1} - \mu)],\end{aligned} \tag{13}$$

where the approximate value of $\overline{K}$ is within a reasonable range. Since $\overline{K}$ is a constant, the first term in the (13) is the variance of $\alpha\overline{K}$. The remaining three terms can be combined into one term based on the linear property. The numerator of the ACF can be simplified as:

$$\begin{aligned}Num(\rho_h) &\approx Var(\alpha\overline{K}) \\ &+ \mathbf{E}[(L_{t+h-1} - \mu)(\beta K_{t-1} - \mu)].\end{aligned} \tag{14}$$

According to the calculation principle of variance, both sides of the (8) is written as:

$$\begin{aligned}Var(L_t) = &Var(\alpha K_t) \\ &+ Var(\beta K_{t-1}) \\ &+ 2Cov(\alpha K_t, \beta K_{t-1}),\end{aligned} \tag{15}$$

where the $Cov()$ is the covariance function. The denominator of (11) can be simplified as:

$$Den(\rho_h) \approx Var(\alpha\overline{K}) + Var(\beta K_{t-1}). \tag{16}$$

Therefore, the ACF formula (11) can be simplified as:

$$\rho_h \approx \frac{Var(\alpha\overline{K}) + \mathbf{E}[(L_{t+h-1} - \mu)(\beta K_{t-1} - \mu)]}{Var(\alpha\overline{K}) + Var(\beta K_{t-1})}. \tag{17}$$

Finally, when the lag order of the ACF is set to 1, the calculation of the ACF can be further simplified under the conditions of recurrence relationships and data similarity, as follows:

$$\rho_1 \approx \frac{Var(\alpha\overline{K})}{Var(\alpha\overline{K}) + Var(\beta K_{t-1})}. \tag{18}$$

Since the first term in the numerator is the same as the denominator, the range of the second term in the denominator determines the range of ρ_1. Besides, the input data x_t that applied to generate the K_t matrix is normalized before being passed to the $Linear()$ function, the variance of the K_t matrix at any given time is determined by the weight matrix. Additionally, the variance of the weight matrices at two consecutive time steps is nearly identical. Therefore, ρ_1 can be expressed as:

$$\rho_1 \approx \frac{\alpha}{\alpha + 1 * \beta}, \tag{19}$$

where the α and β are set to 1.3 and -0.3 respectively. In the presence of Gaussian noise, the ACF result of the ARXSA method is approximately equal to 1. This result indicates that the ARXSA method demonstrates perfect autocorrelation. Therefore, the Encoder block gains more efficiency and accuracy within Transformers robustness(Bhojanapalli et al., 2021).

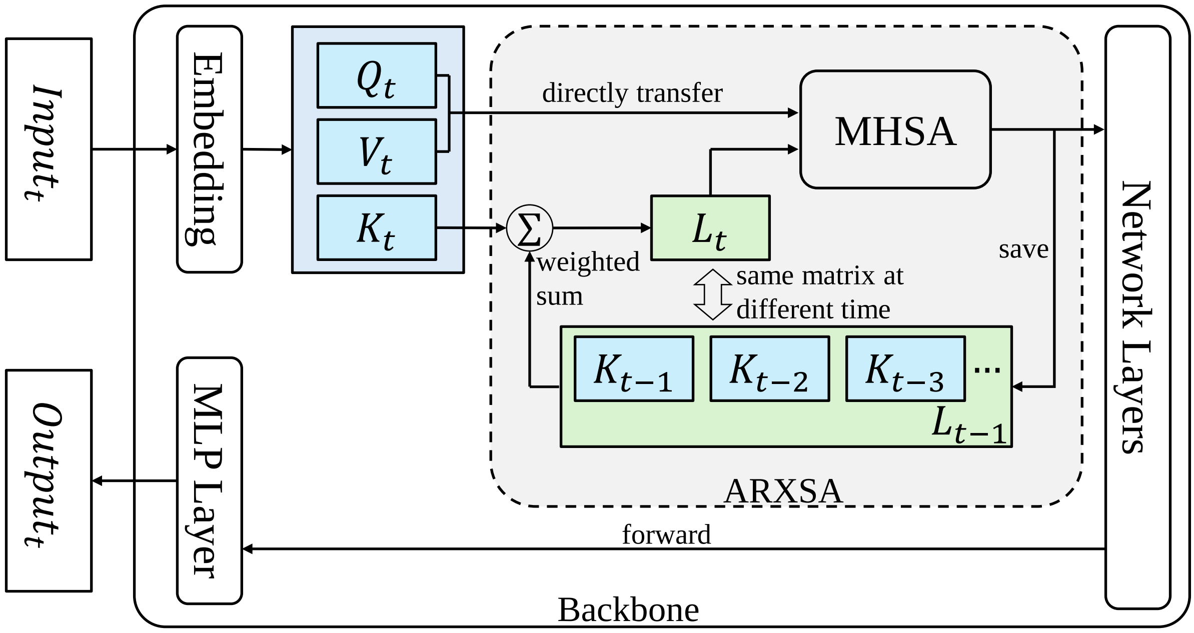

Figure 4: The overall framework of the network with ARXSA method.

2.5 ARXFormer

Based on the above arguments, the ARXSA method is incorporated into the Transformer network to form a neural network with autoregressive properties. Given the high efficiency of patch-based computation, the normal Swin Transformer was selected as the backbone network.

In the network, the interaction of a Stage module with information from adjacent time steps is illustrated in Fig 4. In the ARXFormer with order 1, the input data is transformed into three matrices—Q_t, K_t, and V_t—after passing through the embedding layer. The Q_t and V_t matrices are directly forwarded to the MHSA module for subsequent computation. In contrast, the K_t matrix is first passed through the negative feedback path, where it is combined with L_{t-1} matrix to produce a weighted sum, denoted as L_t. This L_t matrix is then also forwarded to the MHSA module to participate in the self-attention computation alongside the Q_t and Q_t matrices. The resulting attention values are passed to the next network layer, while the input information at time t is stored in the L matrix to be weighted with future information. Finally, the output for the target task is produced after passing through the MLP layer.

3 Experiments

In this section, we selected several popular vision benchmarks to verify the effectiveness of the proposed ARXFormer, including image classification, object detection and text classification. Each set of experiments is validated multiple times and we will evaluate it by the average performance and the single best performance. Since the ARXSA method does not increase the number of parameters, the number of parameters and FLOPs of the same network will not change regardless of whether the ARXSA method is used or not.

3.1 Image Classification

The pre-trained files were applied as base weights in the experiments. Each experiment was set to 50 epochs, with the goal of evaluating the best performance of different methods by comparing the results obtained within a fixed number of epochs. Each batch size was set to 64. To ensure that the experimental results are more representative, three methods were considered: MHSA, Compare, and ARXSA. MHSA represents the multi-head self-attention mechanism, while ARXSA is the proposed method. Compare represents the method of adding a K matrix without interacting information with adjacent batches.

The upper table in Table 1 presents the average precision obtained on different datasets employing the basic Swin Transformer architecture with the three different methods. The lower table in Table 1 shows the highest precision achieved in each set of multiple experiments under the same configuration.

Table 1: Average and highest accuracy of the same neural network utilizing different methods on different datasets.

Method	Flower102	CUB200	CIFAR100	Dog120
MHSA	88.25	70.04	67.48	69.59
Compare	87.63	70.28	67.07	70.34
ARXSA(Ours)	88.54	70.79	68.03	70.40
Method	Flower102	CUB200	CIFAR100	Dog120
MHSA	88.53	71.11	68.17	71.33
Compare	87.73	71.06	67.76	71.45
ARXSA(Ours)	90.00	71.66	68.46	71.36

Similarly, the models equipped with the ARXSA method consistently achieved near-optimal results in both average precision and highest precision. Because the number of parameters and FLOPs do not change under different methods, the comparison of these two parameters is not given. For example, in Flower-102, the number of parameters and FLOPs in the network are 27.57 (M) and 4.37 (G) respectively.

3.2 Object Detection

Object detection subtask experiments were conducted on the VOC dataset, including the 2007 and 2012 versions. MHSA, Compare group and ARXFormer were integrated into the same architecture as the backbone separately(Ren et al., 2015). To test the generalizability of ARXFormer, the settings of each group were always the same. The network employed the SGD optimizer with a weight decay of 0.0005, and pre-trained weights were also employed as initialization parameters.

Table 2: Average and highest results of YOLO utilizing different methods on the same dataset.

Method	mAP	AP_{50}	AP_{75}
MHSA	59.71	72.52	46.88
Compare	59.73	72.60	46.84
ARXSA(Ours)	59.87	72.64	47.12
Method	mAP	AP_{50}	AP_{75}
MHSA	59.90	72.70	47.10
Compare	60.21	72.90	47.50
ARXSA(Ours)	60.02	72.90	47.10

Comparative experiments were conducted employing the YOLO architecture on the VOC dataset(Redmon et al., 2016). To eliminate the influence of different settings within the architecture, all settings in the comparative experiments were kept consistent. The upper table of Table 2 presents the average results of multiple experiments, while another table shows the best results. It can be observed that the architecture equipped with the ARXSA method consistently provides the best performance.

3.3 Text Classification

Text classification subtask experiments were conducted on AG News, IMDb, and DBpedia datasets. BERT was mainly used as the carrier network. Similarly, the presets used in the parts other than the SA module are the same. The pre-trained files used are also the same.

Table 3: Results of BERT utilizing different methods on the same dataset.

Method	AG News	IMDb	DBpedia
MHSA	93.58	88.04	99.07
Compare	87.86	88.09	99.02
ARXSA(Ours)	94.03	88.11	99.07

4 Conclusion

With the recent prevailing trend of the Transformer, we envisioned that integrating negative feedback predictive control theory into the self-attention mechanism to enhance the overall efficiency and stability of the network. Thus, the concept of vertical training and employed it to define the unit time for network training was introduced. Consequently, we proposed and validated the ARXSA method, which transforms the Encoder block in the network into a negative feedback predictive control system. Finally, the ARXSA method was integrated into different network frameworks to verify the applicability and stability. In future work, we will study the performance of ARXSA under different AR orders and study the extension of such methods to learning methods of non-SA mechanisms.

Limitation

The experiments demonstrated that the ARXSA method outperforms MHSA in Swin-like Transformers, and also proved that networks that employing ARXFormer as the backbone exhibit higher stability and computational efficiency. However, ARXSA cannot be applied to ViT-like networks. Since the core matrices in ViT have a high probability of having a determinant of zero, the matrices are almost always non-invertible. As the result, the autocorrelation between core matrices across different batches nearly be zero.

References

Oluleke Babayomi, Zhenbin Zhang, Tomislav Dragicevic, Jiefeng Hu, and Jose Rodriguez. 2023. Smart grid evolution: Predictive control of distributed energy resources—a review. *International Journal of Electrical Power & Energy Systems*, 147:108812.

Srinadh Bhojanapalli, Ayan Chakrabarti, Daniel Glasner, Daliang Li, Thomas Unterthiner, and Andreas Veit. 2021. Understanding robustness of transformers for image classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10231–10241.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. *Advances in neural information processing systems*, 31.

Yuqi Chen, Kan Ren, Yansen Wang, Yuchen Fang, Weiwei Sun, and Dongsheng Li. 2024. Contiformer: Continuous-time transformer for irregular time series modeling. *Advances in Neural Information Processing Systems*, 36.

Jen-Tzung Chien and Yi-Hsiang Chen. 2021. Continuous-time attention for sequential learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7116–7124.

Dick van Dijk, Timo Teräsvirta, and Philip Hans Franses. 2002. Smooth transition autoregressive models—a survey of recent developments. *Econometric reviews*, 21(1):1–47.

Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. 2022. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12124–12134.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, and 1 others. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Willem Esterhuizen, Karl Worthmann, and Stefan Streif. 2020. Recursive feasibility of continuous-time model predictive control without stabilising constraints. *IEEE Control Systems Letters*, 5(1):265–270.

Salvatore Claudio Fanni, Maria Febi, Gayane Aghakhanyan, and Emanuele Neri. 2023. Natural language processing. In *Introduction to Artificial Intelligence*, pages 87–99. Springer.

Aritra Ghosh, Himanshu Kumar, and P Shanti Sastry. 2017. Robust loss functions under label noise for deep neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Xin Gong, Yukang Cui, Jun Shen, Zhiguang Feng, and Tingwen Huang. 2020. Necessary and sufficient conditions of formation-containment control of high-order multiagent systems with observer-type protocols. *IEEE Transactions on Cybernetics*, 52(7):7002–7016.

Salma P González-Sabbagh and Antonio Robles-Kelly. 2023. A survey on underwater computer vision. *ACM Computing Surveys*, 55(13s):1–39.

Ali Hassani, Steven Walton, Jiachen Li, Shen Li, and Humphrey Shi. 2023. Neighborhood attention transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6185–6194.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Yi Huang, Guo-Ping Liu, Yi Yu, and Wenshan Hu. 2024. Data-driven distributed predictive tracking control for heterogeneous nonlinear multi-agent systems with communication delays. *IEEE Transactions on Automatic Control*.

Zhelin Huang, Fan Xu, and Fangfang Yang. 2023. State of health prediction of lithium-ion batteries based on autoregression with exogenous variables model. *Energy*, 262:125497.

Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. 2023. Natural language processing: state of the art, current trends and challenges. *Multimedia tools and applications*, 82(3):3713–3744.

Atli Kosson, Dongyang Fan, and Martin Jaggi. 2024. Ghost noise for regularizing deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13274–13282.

Bonan Li, Yinhan Hu, Xuecheng Nie, Congying Han, Xiangjian Jiang, Tiande Guo, and Luoqi Liu. 2023. Dropkey for vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22700–22709.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022.

Jan M Maciejowski and Mihai Huzmezan. 2007. Predictive control. In *Robust Flight Control: A Design Challenge*, pages 125–134. Springer.

Christoph Mayer, Martin Danelljan, Goutam Bhat, Matthieu Paul, Danda Pani Paudel, Fisher Yu, and Luc Van Gool. 2022. Transforming model prediction for tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8731–8740.

Laurent Meunier, Blaise J Delattre, Alexandre Araujo, and Alexandre Allauzen. 2022. A dynamical system perspective for lipschitz neural networks. In *International Conference on Machine Learning*, pages 15484–15500. PMLR.

Przemysław Podulka, Wojciech Macek, Beata Zima, Grzegorz Lesiuk, Ricardo Branco, and Grzegorz Królczyk. 2023. Roughness evaluation of turned composite surfaces by analysis of the shape of autocorrelation function. *Measurement*, 222:113640.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

Michael L Seltzer, Dong Yu, and Yongqiang Wang. 2013. An investigation of deep neural networks for noise robust speech recognition. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 7398–7402. IEEE.

Nadezhda Semenova, Laurent Larger, and Daniel Brunner. 2022. Understanding and mitigating noise in trained deep neural networks. *Neural Networks*, 146:151–160.

Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. 2021. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3531–3539.

Weixuan Sun, Zhen Qin, Hui Deng, Jianyuan Wang, Yi Zhang, Kaihao Zhang, Nick Barnes, Stan Birchfield, Lingpeng Kong, and Yiran Zhong. 2023. Vicinity vision transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):12635–12649.

Xiaoming Tang, Li Deng, Na Liu, Shuang Yang, and Jimin Yu. 2018. Observer-based output feedback mpc for t–s fuzzy system with data loss and bounded disturbance. *IEEE transactions on cybernetics*, 49(6):2119–2132.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. 2018. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018(1):7068349.

Xiao Ling Wang, Housheng Su, Michael ZQ Chen, Xiao Fan Wang, and Guanrong Chen. 2017. Reaching non-negative edge consensus of networked dynamical systems. *IEEE Transactions on Cybernetics*, 48(9):2712–2722.

Xiaoming Wang, Xinyan Wang, Geyong Min, Fei Hao, and CL Philip Chen. 2020. An efficient feedback control mechanism for positive/negative information spread in online social networks. *IEEE transactions on cybernetics*, 52(1):87–100.

Ee Weinan. 2017. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 1(5):1–11.

Jian Wu, Xuemiao Chen, Qianjin Zhao, Jing Li, and Zheng-Guang Wu. 2020. Adaptive neural dynamic surface control with prespecified tracking accuracy of uncertain stochastic nonstrict-feedback systems. *IEEE Transactions on Cybernetics*, 52(5):3408–3421.

Yuhui Wu, Licai Liu, and Shuqu Qian. 2023. A small sample bearing fault diagnosis method based on variational mode decomposition, autocorrelation function, and convolutional neural network. *The International Journal of Advanced Manufacturing Technology*, 124(11):3887–3898.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and 1 others. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

Jing Zhang, Peng Zhang, Baiwen Kong, Junqiu Wei, and Xin Jiang. 2021. Continuous self-attention models with neural ode networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14393–14401.

Zeyu Zhang, Hongran Li, Heng Zhang, Jian Zhang, Zhaoman Zhong, and Weiwei Xu. 2022. Model-free predictive control of nonlinear systems under false data injection attacks. *Computers and Electrical Engineering*, 100:107977.

Xia Zhao, Limin Wang, Yufei Zhang, Xuming Han, Muhammet Deveci, and Milan Parmar. 2024. A review of convolutional neural networks in computer vision. *Artificial Intelligence Review*, 57(4):99.

Mai Zhu, Bo Chang, and Chong Fu. 2023. Convolutional neural networks combined with runge–kutta methods. *Neural Computing and Applications*, 35(2):1629–1643.