

Article

Not peer-reviewed version

A Study on the Stability of Neural Network Climate Prediction Models with Different Training Stop Criteria

[Xiangjun Shi](#)^{*}, [Ping Zhou](#), Sirui He

Posted Date: 13 April 2026

doi: 10.20944/preprints202604.0821.v1

Keywords: model stability; randomness factors; training stop criteria



Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a [Creative Commons CC BY 4.0 license](#), which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

A Study on the Stability of Neural Network Climate Prediction Models with Different Training Stop Criteria

Xiangjun Shi *, Ping Zhou and Sirui He

School of Atmospheric Sciences, Earth System Modeling Center, Nanjing University of Information Science and Technology, Nanjing, Jiangsu, China

* Correspondence: shixj@nuist.edu.cn

Abstract

Due to randomness factors in the machine learning model construction process, reproducibility is compromised. This study investigates the impact of randomness on model stability and evaluates techniques for reducing this impact, using the widely adopted shallow neural network (NN) model as a testbed. Randomness in this NN model arises from three events: randomly initializing model parameters, randomly selecting a validation subset, and randomly sampling batches for parameter updates. Among these, batch randomness exerts a significantly weaker impact than the other two factors. In this study, the model training is stopped when the validation performance fails to improve or when a preset threshold for loss or epoch-number is met. The final model stability is significantly better using threshold criteria than using validation criterion, as the former avoids the randomness associated with validation subset. Sensitivity experiments show that scaling the model's initial parameters to 0.1 times their original values can mitigate the impact of initialization randomness, thereby significantly improving model stability while also markedly enhancing predictive skill. Furthermore, weight decay and multi-model ensembles, which are two commonly used techniques, can also significantly enhance model stability. Moreover, the inherent instability of individual sub-models may actually benefit the overall predictive skill of a multi-model ensemble.

Keywords: model stability; randomness factors; training stop criteria

1. Introduction

Statistical methods are essential tools for climate prediction, especially for longer lead times [1–3]. Compared to traditional linear statistical methods such as multivariate linear regression, machine learning approaches are increasingly employed in climate prediction research due to their ability to capture complex nonlinear relationships [4–7]. Moreover, ongoing advancements in prediction systems that employ machine learning methods are consistently pushing the predictive skill ceiling upward [8–10].

Unlike most domains in machine learning that benefit from large sample datasets, climate prediction faces the challenge of limited event samples. For most climate events, the observation period is typically too short (less than 100 years) to yield an adequate quantity of samples used for model training. This limitation can be mitigated by pretraining models on data derived from numerical climate model simulations. By expanding the size of training sets, it becomes feasible to adopt relatively complex machine learning models [11–14]. If only observed samples are used, simple machine learning models (i.e., only dozens of parameters) are typically chosen, such as widely applied artificial neural networks (NN) [15,16]. The use of simple models and small dataset minimizes computational costs. This, in turn, facilitates extensive testing and in-depth analysis of issues like model stability. More importantly, this allows for a markedly more efficient development of robust and effective prediction models tailored to climate prediction events [17].

For machine learning models, due to the use of random numbers (i.e., randomness) during the construction process, the final trained model obtained from each run varies (i.e., irreproducibility), resulting in differing model performance. This instability becomes more pronounced in scenarios with a small sample size [18–20]. In recent years, the machine learning community has seen a growing amount of research focused on the small sample size problem. Within this context, some studies have pointed out the impact of randomness. However, the systematic investigation of this issue remains largely underexplored and warrants further dedicated study [21,22]. Furthermore, climate event samples possess an additional characteristic: their quality is low. For instance, two samples may have very similar input predictors, yet their corresponding prediction values can differ significantly. Therefore, research based on climate event samples is particularly valuable, as it can yield more accurate findings for the field of climate prediction.

This study investigates the impact of randomness on model stability and evaluates mitigation techniques, using the prediction of July precipitation in the middle and lower reaches of the Yangtze River (MLYR) based on early winter sea surface temperature (SST) as a case study. In the initial stage, we commence our investigation with a specific shallow NN model that incorporates only three sources of randomness. The paper is structured as follows. Section 2 introduces the NN model and experimental setup. Results and analysis are presented in Section 3. Section 4 discusses how the findings introduced here can be applied, while Section 5 concludes the paper.

2. Data and Methods

2.1. Samples for the NN Model

The original station-observed monthly precipitation data are provided by the National Climate Center of the China Meteorological Administration. First, July precipitation values from all 15 stations in the MLYR region (28.50°N to 32.00°N, 106.48°E to 118.80°E) are averaged into a single value. Next, the decadal trend is removed, precipitation anomaly percentages are calculated, undergo enhancement for abnormal events, and are finally confined to the range of -1 to 1. Hereafter, this indexed precipitation is denoted as Rain. The original monthly SST data are obtained from the IRI/LDEO Climate Data Library [23]. Since the primary indicative winter SST signals originate from the Pacific and Indian Oceans [24,25], winter-mean SST anomalies from the corresponding domain (40°S to 60°N, 30°E to 90°W) are used to extract predictor variables. Following previous studies [15,16,26,27], SST principal components (PCs) serve as potential predictors. From 1951 to 2021, there are a total of 71 climate events (one per year). Each event corresponds to one sample, consisting of one Rain value (i.e., one predictand) and 30 SST PCs (i.e., 30 potential predictors). Appendix A provides more details of the sample data. Furthermore, the data values can be downloaded according to the Data Availability Statement.

2.2. The NN Prediction Model

The prediction model in this study is constructed based on a two-layer NN. The numbers of nodes in the input layer, the first hidden layer, and the second hidden layer are denoted as N_x , N_1 , and N_2 , respectively. The model outputs only one variable (i.e., the predicted Rain). The default model architecture is configured as $N_x=6$, $N_1=8$, $N_2=3$. The number of predictors used in the NN model cannot exceed 30 (i.e., $N_x \leq 30$). Under $N_x < 30$, the PCs are prioritized based on the absolute value of their linear correlation coefficient (Cor) with Rain, with those having higher correlations selected first.

The NN model employs the tanh activation function. At model initialization, all biases are initialized to zero. Weights are initialized with random values drawn from a normal distribution, with the standard deviation set to $1/\sqrt{n}$ (where n denotes the number of nodes in the layer), similar to the commonly used Xavier initialization [28]. During model training, the gradient is computed using the backpropagation algorithm with a batch of randomly selected samples. The batch size is

set to 10. The initial learning rate is set to 0.01, and it is adaptively adjusted throughout the training process based on gradient dynamics via the Adam optimizer [29].

During model training, the model's performance is quantified by a loss function, defined as the Root Mean Squared Error between model predictions and observations (referred to as Loss hereafter). Given the small sample size and the limited predictive power of winter SST for July Rain, excessive model training will inevitably result in overfitting. Considering this, three training stopping criteria are available: Validation, Epoch, and Loss. If the Validation criterion is chosen, the train-dev set (i.e., the raw training set including validation samples) is first randomly split before model training: two-thirds for model training, and the remaining one-third as the validation set. Training is stopped when the loss calculated on the validation set no longer improves. Both the Epoch and Loss are commonly used early stopping criteria, which do not require the validation set [30]. Model training is stopped when the preset threshold is reached. To better compare model performance among these three stopping criteria, the default Loss threshold is set to 0.4 and the default epoch-number threshold to 10. When adjusting the NN architectures or applying stabilization techniques, the default epoch-number threshold may need to be adjusted to ensure that the model performance under the three stopping criteria remains approximately comparable.

The construction process of the NN model can be divided into three independent procedures. Figure 1 illustrates the randomness factors and corresponding random seeds in each procedure. Upon entering each procedure, the randomness factors can be controlled by setting a specific random seed. For instance, a given weight seed corresponds to one initial state of the model. The train-dev set splitting procedure is only applied when the Validation stopping criterion is used. During model training, each parameter update (i.e., each iteration) is based on a batch of randomly sampled samples. The batch seed, which is set before training begins, determines the batch sampling sequence for every iteration within each epoch. Overall, by setting these random seeds, the entire model construction process is reproducible.

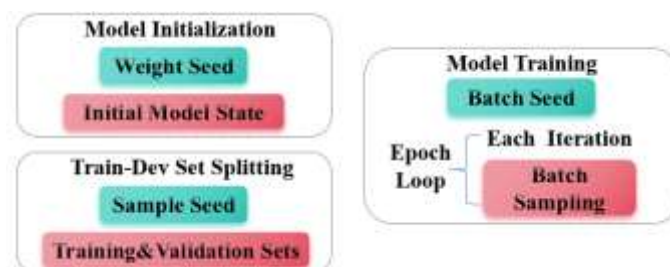


Figure 1. Randomness factors (red box) and corresponding random seeds (green box) under three procedures.

Two simple techniques that enhance stability are introduced into the model: Weight Decay and Initialization0.1. When the Weight Decay technique is applied, all weights and biases are scaled by a decay factor of $(1 - \lambda)$ at each iteration. The decay coefficient λ is set to 0.02. Note that, if the norm of all model parameters falls below a very small threshold (0.1), the Weight Decay mechanism is disabled by setting λ to 0. Conversely, when the norm is large, λ is doubled to enhance its effect. The Initialization0.1 technique is straightforward: it simply scales all weights from the default initialization down to one-tenth of their original values. In addition to the two techniques mentioned above, ensemble experiments are also employed to enhance the stability of the forecast results. This multi-model ensemble method will be introduced in Section 2.3.

To facilitate memorization of the NN model's configuration, Table 1 lists all configuration options relevant to the subsequent experiments. If there are no specific instructions, use the default settings. Moreover, the NN model and all its subroutines (e.g., the random number generator) are coded in Fortran by the authors. Technical details can be found in the source code, which is available for download according to the Data Availability Statement

Table 1. NN model configuration options.

Category	Name	Description (default)
Architecture	Nx	Number of nodes in the input layer (6).
	N1	First hidden layer size (8).
	N2	Second hidden layer size (3).
Training Stop	Validation	No further improvement on validation set.
	Epoch	The number of training epochs is preset (10).
	Loss	Loss falls below a preset threshold (0.4).
Randomness	weight seed	One seed corresponds to one model initial state.
	sample seed	The Train-Validation dataset split is determined.
	batch seed	Controlling randomness during model training.
Stabilization	Weight Decay	Pushes weights and biases toward zero (false).
	Initialization0.1	Initial weights scaled to 0.1x (false).

2.3. Experimental Settings and Evaluation Metrics

The options for the experimental setup are summarized in Table 2. This study conducts two types of experiments: Single and Ensemble. In an Ensemble experimental run, the only difference between individual ensemble members is the random seeds used, which is consistent with the repeated runs in a Single experiment. In this study, each experiment (both Single and Ensemble) comprises 30 repeated runs. By default, the experiments reported are Single experiments. To demonstrate how different ensemble sizes affect model stability, the number of ensemble members is variable and specified for each corresponding experiment.

Table 2. Experimental setup.

Option	Values	Description
Type	Single	The result is the output of each individual NN model.
	Ensemble	The result is the mean value of multiple NN models.
Category	OnlyTrain	All samples are served as Train-Dev (Training) set.
	TrainTest	Partial samples are retained for testing.
	LOO	Leave-One-Out cross-validation experiment.

Given the differences in model's performance between the training and testing sets, three experimental strategies have been designed: OnlyTrain, TrainTest, and LOO. In the OnlyTrain strategy, all samples are used to train the model (including a portion might be reserved as a validation set). In experiments using this strategy, only the model's performance on the train-dev (training) set is provided. Typically, greater attention is placed on the model's performance on the test set, as it provides a more rigorous and generalizable assessment of the model's predictive skill for unseen future events. In the TrainTest strategy, 45 out of the total 71 samples are used as the train-dev (training) set, and 26 as the test set. The results from experiments using this strategy allow for a comparison of the model's performance between the train-dev (training) and test sets. The LOO strategy refers to the well-known Leave-One-Out cross-validation experiment. In each independent run, it selects one sample as the test set and uses the remaining samples as train-dev (training) set, repeating this process until every sample has served as the test set once. Compared to the TrainTest experiment, whose results are subject to the influence of the specific train-dev/test set split, the LOO experiment provides a more robust evaluation of the model predictive skill.

In addition to the Loss, the model's performance is also evaluated using the Cor, a metric commonly employed in climate prediction models. To analyze the model stability, each experiment (both Single and Ensemble types) is repeated 30 times, with the only variation being the random seeds used. Each experimental run produces one prediction model (which can be either a single-model or an ensemble-model), from which a pair of Loss and Cor scores is obtained. The reported Loss and Cor scores are the mean values across all 30 runs, accompanied by their corresponding

standard deviations (presented as mean \pm std.). The standard deviation quantifies (hereafter std.) the dispersion of results across all repeated runs, thereby reflecting the model stability. Furthermore, an additional metric has been designed to quantify the stability (hereafter Sta). For each sample, a standard deviation value can be calculated based on the model-calculated Rain from these 30 runs. The “Sta” score is defined as the average of these standard deviation values across all samples in the dataset. It is evident that a higher Sta value indicates poorer stability. Finally, it is essential to point out that the model stability (instability) discussed in this study specifically refers to the model’s performance (i.e., model-calculated Rain) rather than the model’s parameters (i.e., weights and biases).

3. Results and Analysis

In this section, the impact of randomness factors (i.e., random seeds) on the model construction process is first tested. Next, the relative contribution of each randomness factor to the model instability is assessed. Subsequently, the potential factors influencing model stability are investigated. Finally, techniques for enhancing stability are evaluated.

3.1. Training Process Analysis

Here, the impact of different weight and batch seeds on model training process (including both the initial and final states) is demonstrated through the OnlyTrain experiments with Epoch stopping criterion (Figure 2). During the first two epochs (brown and purple), the model shows significant improvements in the Loss and Cor scores, which is likely due to the learning rate of 0.01 not being particularly small. From epochs 5 (green) to 10 (blue), the model-calculated Rain continues to show improvements in scores. However, a closer look at the Rain values (represented by the green and blue lines) reveals that the improvements are no longer pronounced, as the two lines overlap considerably. This suggests that model improvement progresses only marginally when the epoch-number threshold (10) is approaching.

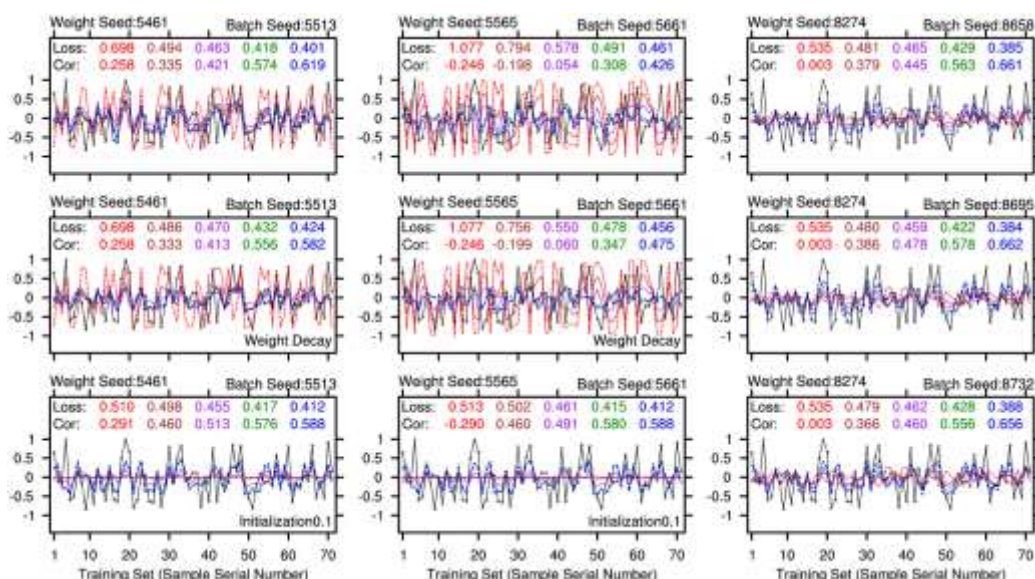


Figure 2. The training processes of nine NN models with varying random seeds, with and without stabilization techniques. The seed values are provided at the top of each plot. The black line denotes the observed samples, while the red, brown, purple, green, and blue lines represent the model performance at the initial time, 1st, 2nd, 5th, and 10th epochs, respectively. The corresponding scores (Loss and Cor) are shown in the matching colors. Two of them use the Initialization0.1, and two use the Weight Decay. The corresponding technical names are labeled at the bottom of the plot.

It is clear that the model initial state is entirely controlled by the weight seed. The initial scores (red color) are identical for a given weight seed. Due to the randomness of the initial states, the Cor scores can be positive or negative. The loss scores are generally not small, with the vast majority being above the standard deviation of the target variable Rain (0.509). With the Initialization0.1 technique (the bottom row), all initial weights are scaled to 0.1 times their original values. As long as the activations remain within the non-saturated (i.e., approximately linear) regime of the tanh function, the signal amplitude will be attenuated to roughly 1/1000 of its original scale after propagating through one input layer and two hidden layers. Therefore, with this technique, the initial model-calculated Rain is near zero. In such a scenario, the initial Loss is essentially the standard deviation of Rain (0.509). This is the reason why the obtained two initial Loss values (0.510 for seed 5461, 0.513 for seed 5565) align closely with each other. While the Initialization0.1 technique compresses the initial model-calculated Rain near zero, the Cor scores (0.291 for seed 5461 and -0.290 for seed 5565) do not change significantly compared to models with the same seeds but without Initialization0.1 (the top row, 0.258 for seed 5461 and -0.246 for seed 5565). This occurs because linear correlation focuses on the consistency of variation trends between the two sequences, rather than their absolute values.

In the first row, the final scores (blue color) of the three models differ significantly (Loss 0.401, 0.461, and 0.385; Cor 0.619, 0.426, and 0.661), likely due to notable differences in their initial states. It appears that a poor initial state likely leads to inferior performance even after 10 epochs of training. The right column of Figure 2 shows the variation in the performances and scores for models trained using weight seed 8274 and three different batch seeds. Although the final (10th-epoch) Loss and Cor scores vary among the three models (Loss 0.385, 0.384, and 0.388; Cor 0.661, 0.662, and 0.656), the differences are not pronounced. Generally speaking, the impact of batch seeds is much less pronounced than that of weight seeds.

The second row shows two model training processes with Weight Decay technique. Compared to the two NN models without Weight Decay (first row, final score Loss 0.401, Cor 0.619; Loss 0.461, Cor 0.426), the difference in final scores between the two sets of random seeds is reduced (Loss 0.424, Cor 0.582; Loss 0.456, Cor 0.475). Due to the Initialization0.1 technique compressing the initial model-calculated Rain near zero, which makes it nearly unaffected by the weight seed, the final scores from the two different sets of random seeds (Loss 0.412, Cor 0.588; Loss 0.412, Cor 0.588) become nearly identical (the bottom row). Overall, both the Weight Decay and Initialization0.1 techniques could improve the stability of the final trained models.

3.2. Contribution of Different Random Seeds to Model Instability

By controlling random seeds as fixed or variable, the impact of a specific random seed on the final trained model can be individually analyzed. Here, the experimental category still adopts OnlyTrain, but all three training stopping criteria are used (Figure 3).

Regardless of the stopping criteria used, the instability induced by batch seeds (first row, Sta 0.058, 0.038, and 0.032) is significantly weaker than that induced by weight seeds (second row, Sta 0.173, 0.147, and 0.137). Among them, the models using the Validation stopping criterion exhibit stronger instability (Sta 0.058 and 0.173) than those employing either the Epoch (Sta 0.038 and 0.147) or Loss stopping criteria (Sta 0.032 and 0.137). One possible reason is that using the validation set to determine when to stop introduces additional complexity into the model training process, which in turn makes it more sensitive to random events. The third row demonstrates how model performance varies with different splits of the train-dev set into training and validation subsets (determined by the sample seed). The model instability induced by sample seeds is obvious, with a comparable magnitude (Sta 0.177) to that from weight seeds (Sta 0.173, second row). The bottom row illustrates the instability contributed by all random seeds. Experiments using both the Epoch and Loss stopping criteria are influenced only by weight seeds and batch seed. The model instability resulting from these two seeds (Sta 0.148 and 0.137) is similar to that induced by weight seeds alone (Sta 0.147 and 0.137). For the Validation stopping criterion, the model instability resulting from all random seeds

(Sta 0.204) is significantly higher than that induced by any single seed type alone (Sta 0.058, 0.173, and 0.177). Overall, the instability induced by validation seeds or weight seeds is significantly stronger than that induced by batch seeds. Furthermore, the model demonstrates higher instability when employing the Validation stopping criterion compared to Epoch or Loss criteria.

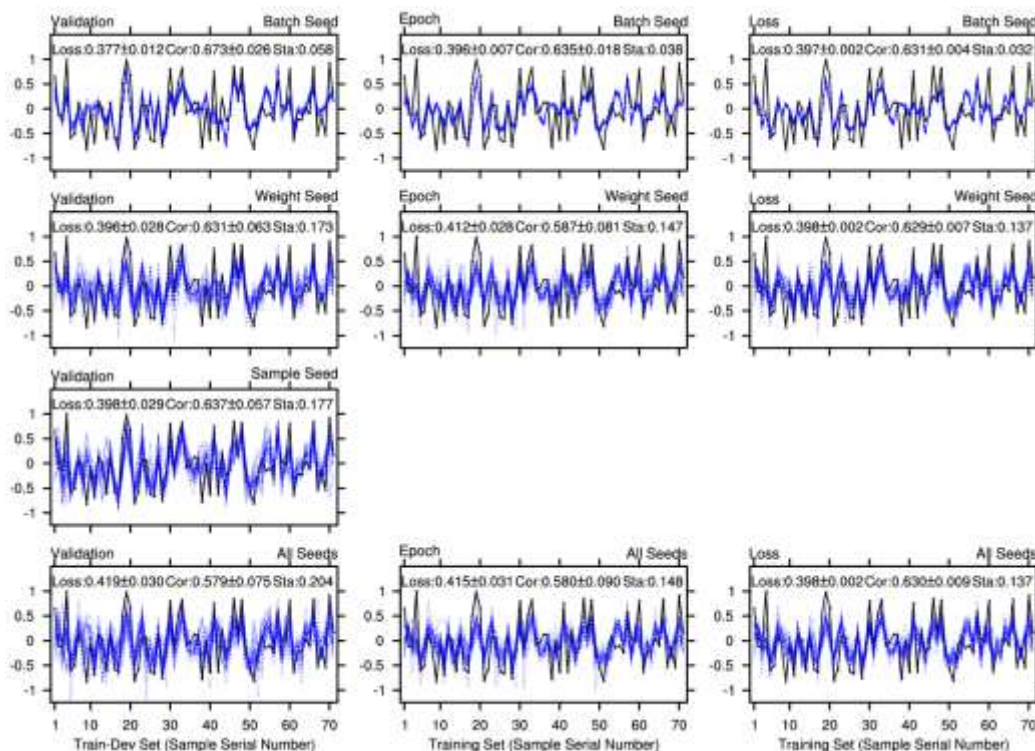


Figure 3. Impact of batch seeds (first row), weight seeds (second row), sample seeds (third row), and all random seeds (fourth row) on the model stability. Each experiment runs 30 times, with only the seeds varying, corresponding to the 30 blue lines in each plot. There are three training stopping criteria: Validation (left), Epoch (middle), and Loss (right).

This paragraph analyzes the stability from the perspective of the std. values of Loss and Cor scores. Under the Validation and Epoch stopping criteria, the std. value generally tracks Sta, rising and falling with it. In other words, the instability of Loss and Cor scores generally covaries with that of the model-calculated Rain (i.e., Sta). However, this rule of thumb does not apply when using the Loss stopping criterion. The use of a Loss stopping criterion (threshold=0.4) truncates the distribution of final Loss scores, artificially clustering them just below 0.4, which naturally results in a very stable Loss score. In the three experiments that used the Loss stopping criterion (first row, second row, and fourth row of right column), the std. values of Loss scores are all 0.002, regardless of whether the model-calculated Rain exhibits significant instability. Since the Cor score is correlated with the Loss score, it remains relatively stable as well. Specifically, the std. values of Cor scores are all below 0.009, which are significantly lower than those from the experiments using the Epoch stopping criterion. It is noteworthy that the Loss stopping criterion contributes to more stable scores on the training set, rather than to more stable model performance (i.e., model-calculated Rain).

3.3. Multi-Perspective Analysis of Model Performance

Here the difference in model stability between the train-dev (training) and test sets is analyzed using TrainTest category with all random seeds varied (Figure 4). Considering the potential impact of different train-dev (training) and test set splits, two distinct dataset partitions are presented here. One partition represents a common or typical scenario (first row), while in the other, the model's scores on the test set is very poor (second row). As expected, the model exhibits noticeably weaker

stability (i.e., higher Sta value) on the test set than on the train-dev (training) set, regardless of the dataset partition or stopping criterion used. Moreover, when test-set performance is particularly poor, as in the second partition, its stability is also substantially worse. Similar to the above analysis in section 3.2, Validation experiments show worse stability than Epoch and Loss experiments, on both the train-dev set and the test set.

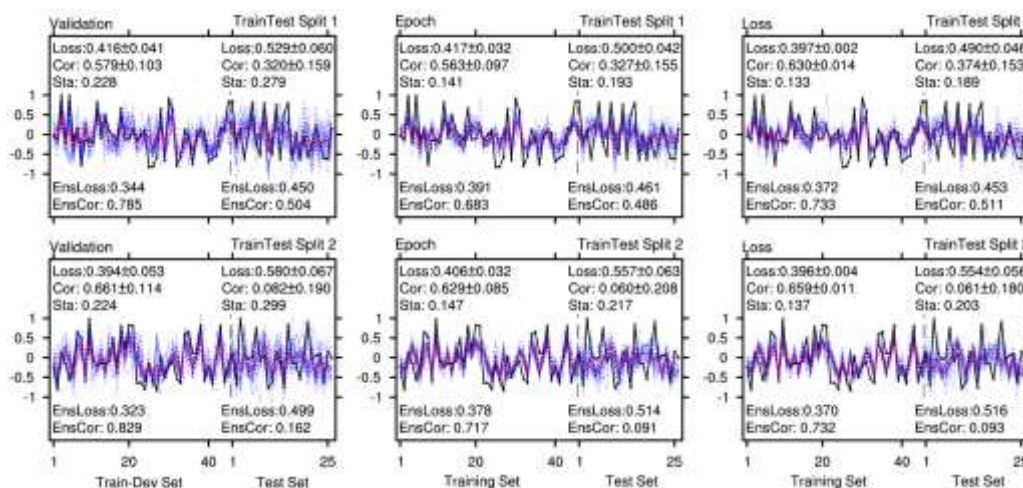


Figure 4. The performance and scores of the NN models from TrainTest experiments with two different train-dev (training) and test set splits (upper and lower panels). The red line in each plot represents the average values of the 30 individual models (i.e., one ensemble experiment), with the corresponding scores annotated at the bottom of each plot.

This paragraph further analyzes the model's performance from the perspective of the Loss and Cor scores. When the Loss stopping criterion is used, although the Loss and Cor scores are stable on the training set, the std. values on the test set increase significantly, approaching the results calculated from experiments using the Epoch stopping criterion. In the first row of the Figure 4, the model's scores on the train-dev set from the Validation experiment (Loss 0.416±0.041; Cor 0.579±0.103) are worse than those from the Loss experiment (Loss 0.397±0.002; Cor 0.630±0.014). However, when analyzed based on the average of the model-calculated Rain from these 30 repeated runs (i.e., as one ensemble experiment), the ensemble-model scores of the Validation experiment (Loss 0.344; Cor 0.785) are better than those of the Loss experiment (Loss 0.372; Cor 0.733). This suggests that the worse stability of single-model (Validation experiments) might be beneficial for achieving better ensemble-model scores. Similarly, although the individual model scores on the test set from the Validation experiment (Loss 0.529±0.060; Cor 0.320±0.159) are worse than those from the Epoch experiment (Loss 0.500±0.042; Cor 0.327±0.155), the ensemble-model scores from the Validation experiment (Loss 0.450; Cor 0.504) are better than the Epoch experiment (Loss 0.461; Cor 0.486). At last, it is noteworthy that ensemble averaging can significantly improve the model's scores.

The influence of model capacity is illustrated using experimental results from two distinct NN architectures (Figure 5): a simple architecture ($N_x=3$, $N_1=4$, $N_2=2$) and a complex architecture ($N_x=20$, $N_1=50$, $N_2=10$). As expected, compared to low-capacity models, high-capacity models exhibit stronger fitting capabilities on the train-dev (training) set but suffer from more severe overfitting issues. In terms of stability (i.e., Sta score), high-capacity models perform significantly worse (higher Sta values) than their low-capacity counterparts, especially on the test set. Low-capacity models maintain similar stability levels on the test and train-dev sets, whereas high-capacity models show a substantial drop in stability on the test set relative to the train-dev set. On the test set, the scores (Loss and Cor) of high-capacity models are significantly worse than those of low-capacity models. However, this conclusion may not hold when evaluated based on ensemble-model scores. Taking Epoch experiments as an example, the ensemble-model score with the complex architecture (Loss

0.466; Cor 0.312) is better than that with the simple architecture (Loss 0.503; Cor 0.285). This again indicates that weaker stability may be beneficial for improving scores via ensemble experiments.

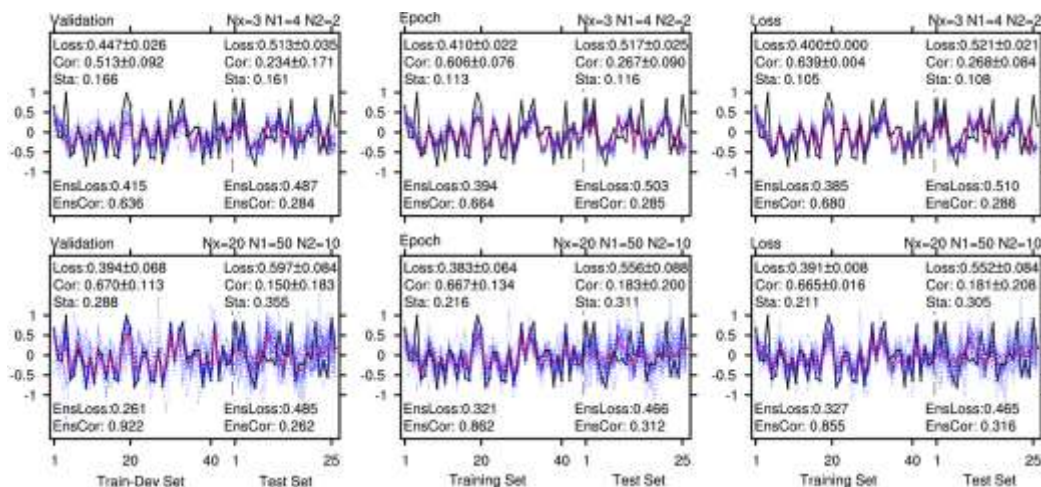


Figure 5. Similar to Figure 4, but for experiments with two different model architectures (indicated in the upper-right corner of each plot). Here, the train-dev (training) and test sets are partitioned according to the temporal sequence of the observed samples.

3.4. Different Techniques for Enhancing Model Stability

Model ensemble is a commonly used technique to enhance predictive reliability as well as mitigate the instability caused by randomness [31]. Since practical applications demand stable performance on future (i.e., test set), we adopt LOO experimental category for a more rigorous evaluation. Figure 6 demonstrates how different ensemble sizes influence model stability. The Loss and Cor corresponding to the three stopping criteria are close to each other. However, the Sta for the Validation stopping criteria is significantly higher than that for the other two stopping criteria (Epoch and Loss). This once again confirms the conclusion drawn from the previous analysis. In the figure, single experiments can be viewed as ensemble experiments with an ensemble size of 1. According to probability theory, if the data follows a normal distribution, then after ensemble averaging, the standard deviation becomes $1/\sqrt{n}$ times the original standard deviation, where n is the ensemble size [32]. For the validation experiments, the Sta values are 0.230, 0.073, 0.041, and 0.020 for ensemble sizes of 1, 10, 30, and 100, respectively. These results are generally consistent with the probability theory introduced above. Note that, from ensemble size=1 to ensemble size=100, the Sta value dropped by more than 10 times, i.e., from 0.230 down to 0.020. For the both Epoch and Loss experiments, the Sta values are also dropped by more than 10 times. One possible reason is that the Sta value here is not the standard deviation on a single sample, but rather the average of standard deviations across all samples. In summary, based on the probability theory introduced above, we can roughly estimate the improvement in stability achieved by increasing ensemble sizes. The actual improvement may exceed this rough estimate.

Figure 7 illustrates the effects of the two techniques: Weight Decay (second row) and Initialization0.1 (third row). To facilitate comparative analysis, the first row shows the reference experiment. It is clear that both the Weight Decay and Initialization0.1 techniques significantly enhance model stability. For experiments using the Epoch or Loss stopping criteria, the Initialization0.1 technique demonstrates remarkable improvement in enhancing model stability. The 30 red lines, each from an ensemble-model output, nearly overlap with one another. This is because the model instability is primarily attributed to its initial states (i.e., weight seeds) and the Initialization0.1 technique reduces the variance among their initial states. For experiments using the Validation stopping criterion, variation in sample seeds is also a major contributor to model instability, which the Initialization0.1 technique cannot eliminate. As a result, the stability from the Validation experiment (Sta 0.040) is worse than that from the Epoch (Sta 0.006) and Loss (Sta 0.008)

experiments. In terms of Loss and Cor, better scores are reported with either Weight Decay or Initialization0.1, regardless of the stopping criteria used. This can be explained by the fact that the Weight Decay technique enhances the model's generalization ability, thereby improving its performance on the test set [33,34]. Similar to the Weight Decay technique, the Initialization0.1 technique also causes the weights in the final model toward smaller magnitudes, thereby enhancing the model's generalization ability (i.e., better predictive skills). In short, both Weight Decay and Initialization0.1 techniques can significantly improve the model's performance, in terms of both stability (i.e., Sta score) and predictive skill (i.e, Loss and Cor scores).

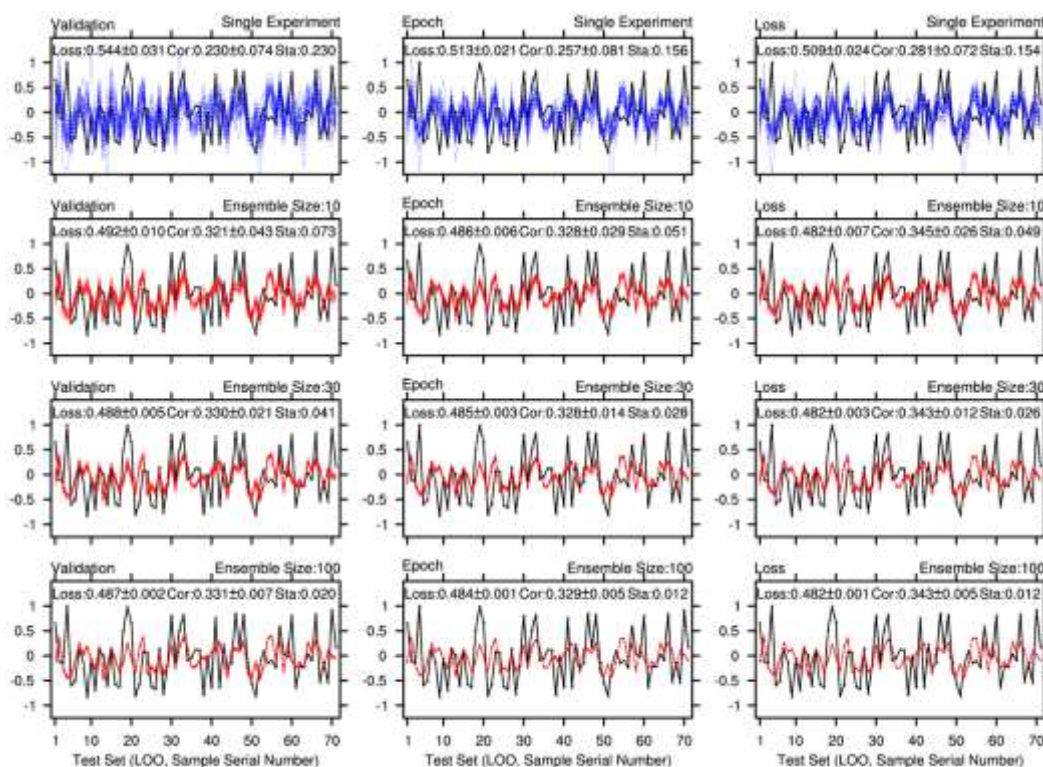


Figure 6. The effect of different ensemble sizes on improving model stability, demonstrated through LOO experiments. Each single model and each ensemble are both run 30 times (represented by 30 blue and 30 red lines, respectively), varying only the random seed. The ensemble sizes are shown in the upper-right corner of each plot.

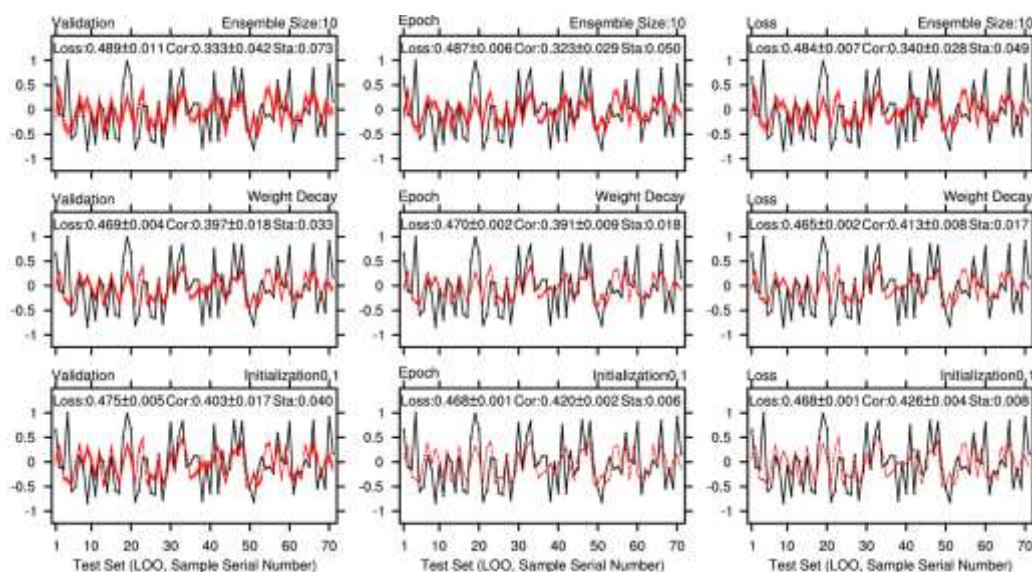


Figure 7. Similar to Figure 6, but for LOO experiments with Weight Decay (second row) and Initialization0.1 (third row). The ensemble size is 10 for all experiments. The first row shows the reference experiments, which have the same configuration as the experiments in the second row of Figure 6, but with different random seeds.

4. Discussion

Based on the experimental analysis presented above, here we discuss practical approaches to building stable and effective NN prediction models. Between the single-model and ensemble-model approaches, it is advisable to prioritize the ensemble-model, as it not only enhances stability but also improves predictive skill. However, it is first necessary to conduct a set of single experiments (e.g., 30 repeated runs) to evaluate the single-model stability (i.e., the Sta value), and then roughly estimate the ensemble size required to achieve an acceptable level of ensemble-model stability. Based on the stable ensemble-model scores, test different model architectures and some related techniques (e.g., Weight Decay), and select the optimal configuration. Among the three stopping criteria: Validation, Epoch, and Loss, try to avoid using Validation. However, when using Epoch or Loss, corresponding thresholds need to be set. These thresholds can be roughly estimated based on the results from Single experiments with the Validation criterion, and then fine-tuned around these initial values to enable the ensemble-model to achieve better predictive skills on the test set. The empirical insights outlined above are primarily applicable to the simple NN model under small-sample climate event scenarios. Their generalizability to other, more complex machine learning models remains uncertain and warrants further investigation.

Dropout is also a widely used regularization technique, which enhances the model's generalization ability [35]. When using dropout, a random selection of neurons are deactivated during each model train iteration. While Dropout is designed to improve generalization, the additional randomness it introduces acts as a destabilizing factor. This instability offsets the performance gains expected from better generalization on the test set, ultimately failing to enhance the model's stability. Unlike benchmark datasets for evaluating AI algorithms (such as MNIST or ImageNet for image classification), where the input and target have an explicit correspondence [36], the relationship between input predictors and the target variable in climate prediction events is associated but not very strong. This might be the primary reason why Dropout does not perform well here. Therefore, the NN model used in this study does not incorporate Dropout, nor are associated experiments carried out.

5. Conclusions

This study investigates the impact of randomness on model stability and evaluates techniques for reducing this impact, based on a case study using a sample NN prediction model. This section will first provide a technical introduction as background and then summarize the research findings.

There are three randomness factors associated with the NN model: randomly initializing model parameters, randomly selecting a validation subset, and randomly sampling batches for parameter updates. These three randomness factors are governed by three random seeds: weight seed, sample seed, and batch seed. By controlling random seeds as fixed or variable, the impact of a specific randomness factor can be individually analyzed. Due to the limited sample size and quality, which poses a high risk of overfitting, three training stopping criteria are provided: Validation, Epoch, and Loss. The Validation criterion means that model training is stopped when the validation performance fails to improve. Unlike the Validation criterion, both the Epoch and Loss criteria do not require a validation set, as they rely solely on reaching a preset threshold. In addition to the commonly used stability-enhancing techniques of weight decay and multi-model ensembles, we propose a novel method to reduce initialization randomness, namely Initialization0.1: scaling the model's initial parameters to 0.1 times their original values.

The experimental results indicate that variations in the sample seed have a comparable impact on model instability to those from the weight seed, while the effect induced by the batch seed is

significantly weaker. When all randomness factors are activated, the final model stability is significantly better using the Epoch or Loss threshold criterion than using the validation criterion, as the former avoids the randomness associated with the sample seed. For the final model trained with the Epoch or Loss threshold criterion, since model instability primarily stems from initialization randomness, the application of the Initialization0.1 technique can yield a very stable ensemble-model with a small ensemble size (10). Furthermore, weight decay and multi-model ensembles, which are two commonly used techniques, can also significantly enhance model stability. Based on the experimental results here, improvements in stability are achieved while predictive skill is also enhanced. When using multi-model ensemble techniques, the weaker stability of individual models is probably beneficial to achieving better predictive skill in the ensemble.

Author Contributions: X.S. designed the study and provided overall support for this work. X.S. developed the neural network model Fortran code and conducted the experiments used in this study. X.S. and P.Z. analyzed the experimental results and drafted the manuscript. All authors have read and approved the final published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (Grant 42475174). The APC was funded by the same funder.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The materials for this study, which comprise the Fortran code (utilizing observed samples as read-in data) and the experimental results (including plotting scripts and raw figures), have been archived in a public repository: <https://doi.org/10.5281/zenodo.19468997> (accessed on April 8, 2026).

Acknowledgments: This study was conducted in the High-Performance Computing Center of Nanjing University of Information Science & Technology. During the preparation of this manuscript, the authors used DeepSeek (online version, continuously updated) for the purposes of checking English writing grammar and moderately polishing the logical expression based on some revision suggestions.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Figure A shows the 15 stations in the MLYR region. It is necessary to point out that, among these stations, the composite maps of prophase SST for years with high and low precipitation anomalies exhibit good similarity. A 21-point (year) moving average is applied to remove the decadal trend and calculate the precipitation anomaly percentage. Years with precipitation anomaly percentages exceeding $\pm 25\%$ are defined as abnormal years. The precipitation anomaly percentages for abnormal years are amplified by 25% (e.g., 40% becomes 65% and -30% becomes -55%). This increases their contribution in establishing the prediction model, thereby strengthening the model's capacity to predict abnormal events. The EOF analysis is first performed on the winter SST anomalies from 1951 to 2021 to derive the 30 leading eigenvectors. The 30 PCs are then calculated by projecting the SST anomalies onto these EOF eigenvectors. Note that the perturbation magnitudes vary across different PCs, and normalization is not used. Those 30 PCs are ranked in descending order of their correlation with Rain. In the default configuration of the NN prediction model, only the first 6 PCs are used as input predictors. Figure A shows the six EOF modes used for calculating SST predictor variables (i.e., PCs) and the specific data of observed samples.

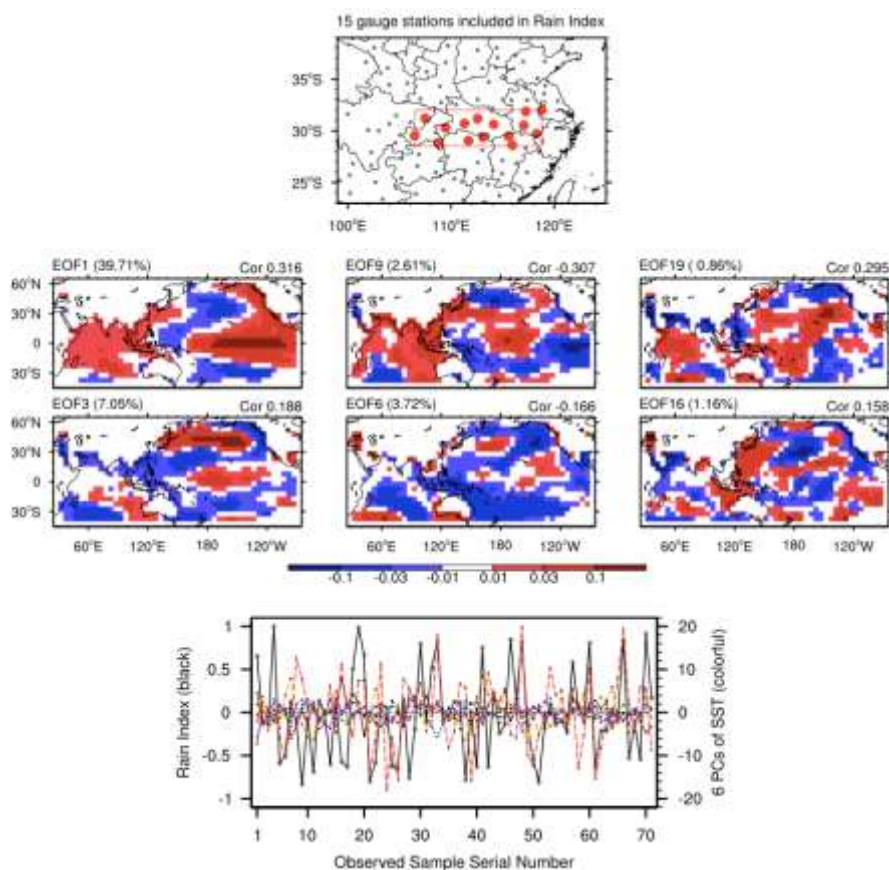


Figure A1. The rain gauge stations (red dots) used for calculating the Rain Index (upper panel), the six EOF eigenvectors corresponding to the six SST PCs (default setting) that serve as predictor variables (middle panel), and the observed samples (one Rain Index and six SST PCs) used for the NN model (lower panel). The EOF mode number and its variance contribution are shown at the upper-left corner, and the linear correlation coefficient (Cor) between the corresponding PC and Rain is shown at the upper-right corner. For the observed samples, the Rain Index is plotted as a black line (left Y-axis), while the six SST PCs are plotted as colorful lines (right Y-axis).

References

1. Liu, Y.; Ren, H.; Zhang, P.; Zuo, J.; Tian, B.; Wan, J.; Li, Y. Application of the Hybrid Statistical Downscaling Model in Summer Precipitation Prediction in China. *Clim. Environ. Res.* **2020**, *25*, 163–171. <https://doi.org/10.3878/j.issn.1006-9585.2019.18168>.
2. Troccoli, A. Seasonal Climate Forecasting. *Meteorol. Appl.* **2010**, *17*, 251–268. <https://doi.org/10.1002/met.184>.
3. Toride, K.; Newman, M.; Hoell, A.; Capotondi, A.; Schlör, J.; Amaya, D.J. Using Deep Learning to Identify Initial Error Sensitivity for Interpretable ENSO Forecasts. *Artif. Intell. Earth Syst.* **2025**, *4*, e240045. <https://doi.org/10.1175/AIES-D-24-0045.1>.
4. Wang, T.; Huang, P. Superiority of a Convolutional Neural Network Model over Dynamical Models in Predicting Central Pacific ENSO. *Adv. Atmos. Sci.* **2024**, *41*, 141–154. <https://doi.org/10.1007/s00376-023-3001-1>.
5. Mekanik, F.; Imteaz, M.A.; Gato-Trinidad, S.; Elmahdi, A. Multiple Regression and Artificial Neural Network for Long-Term Rainfall Forecasting Using Large Scale Climate Modes. *J. Hydrol.* **2013**, *503*, 11–21. <https://doi.org/10.1016/j.jhydrol.2013.08.035>.
6. Anirudh, K.M.; Raj, P.; Sandeep, S.; Kodamana, H.; Sabeerali, C.T. A Skillful Prediction of Monsoon Intraseasonal Oscillation Using Deep Learning. *J. Geophys. Res. Mach. Learn. Comput.* **2025**, *2*, e2024JH000504. <https://doi.org/10.1029/2024JH000504>.

7. Bommer, P.L.; Kretschmer, M.; Spuler, F.R.; Bykov, K.; Höhne, M.M.-C. Deep Learning Meets Teleconnections: Improving S2S Predictions for European Winter Weather. *Mach. Learn. Earth* **2025**, *1*, 015002. <https://doi.org/10.1088/3049-4753/ade9c2>.
8. Yang, S.; Ling, F.; Ying, W.; Yang, S.; Luo, J. A Brief Overview of the Application of Artificial Intelligence to Climate Prediction. *Trans. Atmos. Sci.* **2022**, *45*, 641. <https://doi.org/10.13878/j.cnki.dqkxxb.20210623003>.
9. Wang, G.-G.; Cheng, H.; Zhang, Y.; Yu, H. ENSO Analysis and Prediction Using Deep Learning: A Review. *Neurocomputing* **2023**, *520*, 216–229. <https://doi.org/10.1016/j.neucom.2022.11.078>.
10. Materia, S.; García, L.P.; van Straaten, C.; O, S.; Mamalakis, A.; Cavicchia, L.; Coumou, D.; de Luca, P.; Kretschmer, M.; Donat, M. Artificial Intelligence for Climate Prediction of Extremes: State of the Art, Challenges, and Future Perspectives. *WIREs Clim. Change* **2024**, *15*, e914. <https://doi.org/10.1002/wcc.914>.
11. Ham, Y.-G.; Kim, J.-H.; Luo, J.-J. Deep Learning for Multi-Year ENSO Forecasts. *Nature* **2019**, *573*, 568–572. <https://doi.org/10.1038/s41586-019-1559-7>.
12. Gibson, P.B.; Chapman, W.E.; Altinok, A.; Delle Monache, L.; DeFlorio, M.J.; Waliser, D.E. Training Machine Learning Models on Climate Model Output Yields Skillful Interpretable Seasonal Precipitation Forecasts. *Commun. Earth Environ.* **2021**, *2*, 159. <https://doi.org/10.1038/s43247-021-00225-4>.
13. Yang, S.; Ling, F.; Li, Y.; Luo, J.-J. Improving Seasonal Prediction of Summer Precipitation in the Middle–Lower Reaches of the Yangtze River Using a TU-Net Deep Learning Approach. *Artif. Intell. Earth Syst.* **2023**, *2*, 220078. <https://doi.org/10.1175/AIES-D-22-0078.1>.
14. Hoffman, L.; Massonnet, F.; Sticker, A. Probabilistic Forecasts of September Arctic Sea Ice Extent at the Interannual Timescale With Data-Driven Statistical Models. *J. Geophys. Res. Mach. Learn. Comput.* **2025**, *2*, e2025JH000669. <https://doi.org/10.1029/2025JH000669>.
15. Tangang, F.T.; Hsieh, W.W.; Tang, B. Forecasting the Equatorial Pacific Sea Surface Temperatures by Neural Network Models. *Clim. Dyn.* **1997**, *13*, 135–147. <https://doi.org/10.1007/s003820050156>.
16. Wu, A.; Hsieh, W.W.; Tang, B. Neural Network Forecasts of the Tropical Pacific Sea Surface Temperatures. *Neural Netw.* **2006**, *19*, 145–154. <https://doi.org/10.1016/j.neunet.2006.01.004>.
17. Sun, C.; Shi, X.; Yan, H.; Jiang, Q.; Zeng, Y. Forecasting the June Ridge Line of the Western Pacific Subtropical High with a Machine Learning Method. *Atmosphere* **2022**, *13*, 660. <https://doi.org/10.3390/atmos13050660>.
18. Amir, S.; Van De Meent, J.-W.; Wallace, B. On the Impact of Random Seeds on the Fairness of Clinical Classifiers. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, 6–11 June 2021.
19. Gundersen, O.E.; Coakley, K.; Kirkpatrick, C.; Gil, Y. Sources of Irreproducibility in Machine Learning: A Review. *arXiv* **2023**, *arXiv:2204.07610*. <https://doi.org/10.48550/arXiv.2204.07610>.
20. Zhang, T.; Wu, F.; Katiyar, A.; Weinberger, K.Q.; Artzi, Y. Revisiting Few-Sample BERT Fine-Tuning. *arXiv* **2021**, *arXiv:2006.05987*. <https://doi.org/10.48550/arXiv.2006.05987>.
21. Keshari, R.; Ghosh, S.; Chhabra, S.; Vatsa, M.; Singh, R. Unravelling Small Sample Size Problems in the Deep Learning World. In Proceedings of the 2020 IEEE Sixth International Conference on Multimedia Big Data (BigMM), New Delhi, India, 24–26 September 2020. <https://doi.org/10.1109/BigMM50055.2020.00028>.
22. Pecher, B.; Srba, I.; Bielikova, M. A Survey on Stability of Learning with Limited Labelled Data and Its Sensitivity to the Effects of Randomness. *ACM Comput. Surv.* **2025**, *57*, 1–40. <https://doi.org/10.1145/3691339>.
23. Kaplan, A.; Cane, M.A.; Kushnir, Y.; Clement, A.C.; Blumenthal, M.B.; Rajagopalan, B. Analyses of Global Sea Surface Temperature 1856–1991. *J. Geophys. Res. Oceans* **1998**, *103*, 18567–18589. <https://doi.org/10.1029/97JC01736>.
24. Fan, K.; Wang, H.; Choi, Y.-J. A Physically-Based Statistical Forecast Model for the Middle-Lower Reaches of the Yangtze River Valley Summer Rainfall. *Chin. Sci. Bull.* **2008**, *53*, 602–609. <https://doi.org/10.1007/s11434-008-0083-1>.
25. Huang, P.; Huang, R.-H. Relationship between the Modes of Winter Tropical Pacific SST Anomalies and the Intraseasonal Variations of the Following Summer Rainfall Anomalies in China. *Atmos. Ocean. Sci. Lett.* **2009**, *2*, 295–300. <https://doi.org/10.1080/16742834.2009.11446813>.

26. Cai, Y.; Shi, X. A Comparative Study on the Methods of Predictor Extraction from Global Sea Surface Temperature Fields for Statistical Climate Forecast System. *Atmosphere* **2025**, *16*, 349. <https://doi.org/10.3390/atmos16030349>.
27. Jiang, Q.; Shi, X. Forecasting the July Precipitation over the Middle-Lower Reaches of the Yangtze River with a Flexible Statistical Model. *Atmosphere* **2023**, *14*, 152. <https://doi.org/10.3390/atmos14010152>.
28. Patel, P.; Nandu, M.; Raut, P. Initialization of Weights in Neural Networks. *Int. J. Sci. Dev. Res.* **2018**, *3*, 73–79.
29. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, *arXiv:1412.6980*. <https://doi.org/10.48550/arXiv.1412.6980>.
30. Mahsereci, M.; Balles, L.; Lassner, C.; Hennig, P. Early Stopping without a Validation Set. *arXiv* **2017**, *arXiv:1703.09580*. <https://doi.org/10.48550/arXiv.1703.09580>.
31. Gundersen, O.E.; Shamsaliei, S.; Kjærnl, H.S.; Langseth, H. On Reporting Robust and Trustworthy Conclusions from Model Comparison Studies Involving Neural Networks and Randomness. In Proceedings of the 2023 ACM Conference on Reproducibility and Replicability, Santa Cruz, CA, USA, 28 June 2023. <https://doi.org/10.1145/3589806.3600044>.
32. Sheng, Z.; Xie, S.; Pan, C. *Probability and Mathematical Statistics*; Higher Education Press: Beijing, China, 2008; pp. 30–54. (In Chinese)
33. Krogh, A.; Hertz, J. A Simple Weight Decay Can Improve Generalization. In *Advances in Neural Information Processing Systems 4*; Moody, J.E., Hanson, S.J., Lippmann, R.P., Eds.; Morgan Kaufmann: San Mateo, CA, USA, 1992; pp. 950–957.
34. Smith, L.N. A Disciplined Approach to Neural Network Hyper-Parameters: Part 1 -- Learning Rate, Batch Size, Momentum, and Weight Decay. *arXiv* **2018**, *arXiv:1803.09820*. <https://doi.org/10.48550/arXiv.1803.09820>.
35. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
36. Mamalakis, A.; Ebert-Uphoff, I.; Barnes, E.A. Neural Network Attribution Methods for Problems in Geoscience: A Novel Synthetic Benchmark Dataset. *Environ. Data Sci.* **2022**, *1*, e8. <https://doi.org/10.1017/eds.2022.7>.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.