CAWI: Copula-Aligned Weight Initialization for Randomized Neural Networks

Mushir Akhtar **M. Tanveer** **Mohd. Arshad**
Indian Institute of Technology Indore, Simrol, Indore, 453552, India

Abstract

Randomized neural networks (RdNNs) enable efficient, backpropagation-free training by freezing randomly initialized input-to-hidden weights, which permits a closed-form solution for the output layer. However, conventional random initialization is blind to inter-feature dependence—ignoring correlations, asymmetries, and tail dependence in the data—which degrades conditioning and predictive performance. To the best of our knowledge, this limitation remains unaddressed in the RdNN literature. To close this gap, we propose CAWI (Copula-Aligned Weight Initialization), a framework that draws input-to-hidden weights from a data-fitted copula that matches empirical dependence, ensuring the frozen projections respect inter-feature dependence without sacrificing closed-form solution. CAWI (i) maps each feature to the unit interval using empirical CDFs, (ii) fits a multivariate copula that captures rank-based dependence among features, and (iii) samples each weight column w_j from the fitted copula and applies a fixed inverse marginal transform to set scale. The objective, solver, and "freeze-once" paradigm remain unchanged; only the sampling law for W becomes dependence-aware. For dependence modeling, we consider two copula families: elliptical (Gaussian, t) and Archimedean (Clayton, Frank, Gumbel). This enables CAWI to handle diverse dependence, including tail dependence. We evaluate CAWI across 83 diverse classification benchmarks (binary and multiclass) and two biomedical datasets, BreaKHis and the Schizophrenia dataset, using standard shallow and deep RdNN architectures. CAWI consistently delivers significant improvements in predictive performance over conventional random initialization. Codes are provided at `https://github.com/mtanveer1/CAWI`.

Proceedings of the 29[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

1 Introduction

Deep learning (DL) models have achieved remarkable success across many domains, from image recognition Voulodimos et al. (2018); Chen et al. (2019); Liu et al. (2024) to natural language processing Otter et al. (2021); Lauriola et al. (2022); Yin et al. (2024), due to their ability to learn rich hierarchical representations of data LeCun et al. (2015); Luo et al. (2023). End-to-end training with backpropagation refines millions of parameters to minimize task-specific losses, yielding expressive features and strong generalization. However, state-of-the-art performance typically demands substantial computation, broad hyperparameter search, and long training cycles Goodfellow (2016); Tiwari et al. (2023). Furthermore, training deep architectures can be hindered by vanishing or exploding gradient Pascanu et al. (2013); Jaiswal et al. (2022); Ceni (2025).

The aforementioned limitations of DL models have prompted researchers to seek alternative neural architectures that couple competitive predictive performance with markedly lower training cost. A prominent family is randomized neural networks (RdNNs) Pao et al. (1994); Cao et al. (2018); Suganthan and Katuwal (2021); Hu et al. (2024). In RdNNs, a substantial subset of parameters—typically input-to-hidden weights—are sampled once from simple distributions and then held fixed, while the remaining (often output) weights are obtained by solving a single linear system, commonly with Tikhonov regularization, instead of iterative backpropagation Zhang and Suganthan (2016). This paradigm eliminates the need for iterative weight updates, substantially reducing training time and computational overhead, yet still allows RdNNs to learn complex input-output mappings Zhang and Suganthan (2016). Theoretically, RdNNs satisfy universal approxi-

mation on compact domains, approximating any continuous function arbitrarily well given sufficient width and appropriate activations Igelnik and Pao (1995); Needell et al. (2024).

In recent years, RdNNs have witnessed significant advancements, particularly in methods aimed at improving their robustness and scalability. Ensemble-based approaches have been introduced to enhance generalization and mitigate variability Shi et al. (2021); Guo et al. (2021). Granular ball-based scalable methods have further extended the applicability of RdNNs to large-scale and high-dimensional datasets Sajid et al. (2025a). Additionally, the integration of fuzzy inference systems into RdNN frameworks has provided a way to incorporate uncertainty handling and interpretability, making them more effective for complex and imprecise decision-making tasks Sajid et al. (2024b, 2025b). Beyond these trends, a number of technical advances have broadened the RdNN toolkit Wong et al. (2022); Chen et al. (2025); Akhtar et al. (2025); Ren et al. (2025).

Despite several advancements, RdNNs face a fundamental limitation rooted in the very mechanism that makes them efficient: the input-to-hidden weights are randomly initialized once and then held fixed throughout training. With the matrix $W \in \mathbb{R}^{d\times h}$ frozen, the induced feature map $x \mapsto \phi(xW+b)$ is non-adaptive; the hidden representation $H = \phi(XW + \mathbf{1}_m b^\top) \in \mathbb{R}^{m\times h}$ is therefore determined by this random draw rather than learned from data. The fixed representation H cannot adjust to dependencies among inputs (e.g., correlations, relative scales, higher-order interactions), so at practical hidden dimension h it may underrepresent the joint patterns that drive prediction. The readout then computes $\hat{Y} = H\Theta$ with $\Theta \in \mathbb{R}^{h\times n}$; if relevant dependencies are absent from H, they cannot be recovered, leading to persistent approximation bias and lower performance at a given model size.

The preceding limitation naturally raises a question: can we make the frozen projections dependence-aware without abandoning closed-form training? In this work, we answer this question by introducing CAWI (Copula-Aligned Weight Initialization), a method that uses copula theory to generate *dependence-aware* input-to-hidden weights. Copulas model multivariate data by separating the univariate marginals from the joint dependence structure, allowing us to learn dependence without committing to specific marginal forms Nelsen (2006); Joe (2014). This separation is well suited to tabular learning, where feature scales and marginals vary widely but predictive signal often resides in cross-feature structure. Recent literature demonstrates broad utility of copulas in AI: copulas for time-series forecasting Sun and Yu (2022); Ashok et al. (2024), interpretable estimation of CNN deep-feature densities using copulas and the generalized characteristic function Chapman and Farvardin (2024), and survival analysis under dependent censoring with identifiability guarantees Zhang et al. (2024). Further developments include copula-nested spectral kernel networks Tian et al. (2024), scalable mixed models with arbitrary marginals Simchoni and Rosset (2025), and neural copula density estimation Letizia et al. (2025), among others.

Several works in deep neural networks have explored structured or data-aware initialization strategies, such as orthogonal weight initialization Hu et al. (2020); Narkhede et al. (2022) and whitening-based transformations LeCun et al. (2002); Kessy et al. (2018). Orthogonal schemes stabilize signal propagation via algebraic constraints but remain distribution-agnostic, while whitening decorrelates features through covariance normalization, primarily addressing linear dependencies. More generally, data-dependent projection methods modify the feature space (e.g., via PCA or kernel mappings), often at additional computational cost.

In contrast, CAWI is designed for RdNNs, where hidden weights are sampled once and kept fixed. Rather than altering the feature representation or imposing structural constraints, CAWI modifies only the sampling distribution of the weight matrix using an empirical copula fitted to the training features. Specifically, we construct rank-based pseudo-observations, estimate elliptical (Gaussian, t) or Archimedean (Clayton, Frank, Gumbel) copulas, and draw each column of W from the fitted dependence structure with fixed marginal scaling. The resulting frozen projections inherit empirical inter-feature dependence, while the RdNN architecture, activation functions, objective, and closed-form solver remain unchanged, incurring essentially no additional training cost.

Our Contributions

- We introduce CAWI (Copula-Aligned Weight Initialization), a plug-in scheme that makes randomized neural networks dependence-aware by sampling the columns of W from a copula fitted to the training features, while preserving the closed-form solver and the freeze-once paradigm.
- We provide numerically stable constructions for elliptical (Gaussian, t) and Archimedean (Clayton, Frank, Gumbel) copulas via rank-based estimation and nearest-SPD projection.
- We evaluate CAWI on 83 diverse classification benchmarks (binary and multiclass) and two biomedical datasets (BreaKHis and the Schizophrenia dataset) using both shallow and deep RdNN architectures, and observe consistent improvements,

with 4–5% relative gains on some datasets.

2 Preliminaries

2.1 Randomized neural network (RdNN) setting

Let $X \in \mathbb{R}^{m\times d}$ be the input matrix (m samples, d features) and $Y \in \mathbb{R}^{m\times n}$ the target matrix (e.g., one–hot labels). An RdNN draws a single hidden layer of h units by sampling a *frozen* weight matrix $W \sim \mathcal{D}_W \subset \mathbb{R}^{d\times h}, b \in \mathbb{R}^h$, from a prescribed base distribution (classically i.i.d. uniform or Gaussian, independent across coordinates), and then *never updates* these parameters. With an elementwise nonlinearity $\phi : \mathbb{R}\to\mathbb{R}$, the hidden representation is

$$H \;=\; \phi\big(XW + \mathbf{1}_m b^\top\big) \;\in\; \mathbb{R}^{m\times h}, \tag{1}$$

where $\mathbf{1}_m$ denotes an m–vector of ones broadcasting the bias. Training reduces to fitting only the output weights $\Theta \in \mathbb{R}^{h\times n}$ by ridge regression:

$$\Theta^\star \;\in\; \arg\min_{\Theta\in\mathbb{R}^{h\times n}} \|H\Theta - Y\|_F^2 + \frac{\lambda}{2}\|\Theta\|_F^2, \qquad \lambda \ge 0, \tag{2}$$

which admits the standard closed forms

$$\text{(primal)}\quad \Theta^\star = (H^\top H + \lambda I_h)^{-1} H^\top Y, \tag{3}$$

$$\text{(dual)}\quad \Theta^\star = H^\top (HH^\top + \lambda I_m)^{-1} Y. \tag{4}$$

Thus *no backpropagation is required*: once W and b are sampled, training is a single linear solve. Detailed formulation and architecture of standard RdNN models are discussed in Section A of the Appendix.

2.2 Copulas

Copulas are a classical device for isolating and modeling *dependence* separately from *marginals* Nelsen (2006); Joe (2014). Sklar's seminal result shows that any multivariate distribution can be written as a coupling of its univariate marginals through a single function on the unit hypercube, thereby reducing multivariate modeling to estimating marginals and a dependence map on $[0,1]^d$ Sklar (1959).

Copula: Let $\mathbf{X} = (X_1, \ldots, X_d)$ be a random vector with marginal cumulative distribution functions (CDFs) $F_j(t) = \Pr[X_j \le t]$ for $j = 1, \ldots, d$. The *copula associated with* $\mathbf{X}$ is the joint CDF of the transformed vector $(F_1(X_1), \ldots, F_d(X_d)) \in [0,1]^d$, namely

$$C(u_1, \ldots, u_d) = \Pr\big\{F_1(X_1) \le u_1, \ldots, F_d(X_d) \le u_d\big\}, \quad (u_1, \ldots, u_d) \in [0,1]^d. \tag{5}$$

Equivalently, a d–variate copula is a distribution function on $[0,1]^d$ whose univariate marginals are uniform on $(0,1)$. Figure 1 provides an intuitive picture of the concept.

Invariance: If ψ_j are strictly increasing for each j, then $(\psi_1(X_1), \ldots, \psi_d(X_d))$ has the same copula as $(X_1, \ldots, X_d)$. Thus a copula depends only on cross-coordinate order, not on units or scales.

Sklar's theorem: Let $F : \mathbb{R}^d \to [0,1]$ be the joint CDF of $\mathbf{X} = (X_1, \ldots, X_d)$ with marginals $F_1, \ldots, F_d$. There exists a copula C such that, for all $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$,

$$F(x_1, \ldots, x_d) \;=\; C\big(F_1(x_1), \ldots, F_d(x_d)\big).$$

If each F_j is continuous, the copula C is unique. Conversely, for any copula C and any marginals $F_1, \ldots, F_d$, the mapping above defines a valid d-variate distribution. If $X_1, \ldots, X_d$ are mutually independent, the copula is the *product copula* $C(u_1, \ldots, u_d) = \prod_{j=1}^d u_j$, representing absence of dependence among coordinates.

3 Problem Statement

Randomized neural networks (RdNNs) *sample once and freeze* the input-to-hidden weight matrix $W \in \mathbb{R}^{d\times h}$ and bias $b \in \mathbb{R}^h$. With $H = \phi(XW + \mathbf{1}_m b^\top)$ and $\Theta^\star(W) = \arg\min_\Theta\{\|H\Theta - Y\|_F^2 + \lambda\|\Theta\|_F^2\}$, training reduces to a single ridge-regularized linear solve—no backpropagation is required.

Limitation. The longstanding bottleneck is how W is drawn: almost universally from an *i.i.d.*, factorized base law (uniform or Gaussian). Equivalently, this imposes the *product copula* $C_\perp(u) = \prod_{j=1}^d u_j$ across weight coordinates, irrespective of the empirical dependence in the inputs X (correlations, asymmetries, tail dependence). The resulting mismatch can (i) reduce the diversity/effective rank of H, (ii) fail to excite discriminative co-movements, and (iii) worsen the conditioning of $H^\top H + \lambda I$.

Goal. Replace the independence assumption on frozen weights *without* altering the RdNN pipeline. Given data $X \in \mathbb{R}^{m\times d}$, width h, activation ϕ, and ridge λ, training remains the same closed form; only the *sampling law* for W changes. We seek a *one-shot, data-aware* sampler $\mathcal{S}$ that maps X to a distribution over W such that each column $w \in \mathbb{R}^d$ has a joint law whose *copula* matches an estimator of input dependence, while its marginals are simple and well-behaved:

$$\text{Copula}(w) \approx \widehat{\text{Copula}}(X), w_j \sim G \text{ for a fixed marginal } G \text{ (e.g., } \mathcal{N}(0,1) \text{ or } \mathcal{U}[-1,1]),\ j = 1, \ldots, d. \tag{6}$$

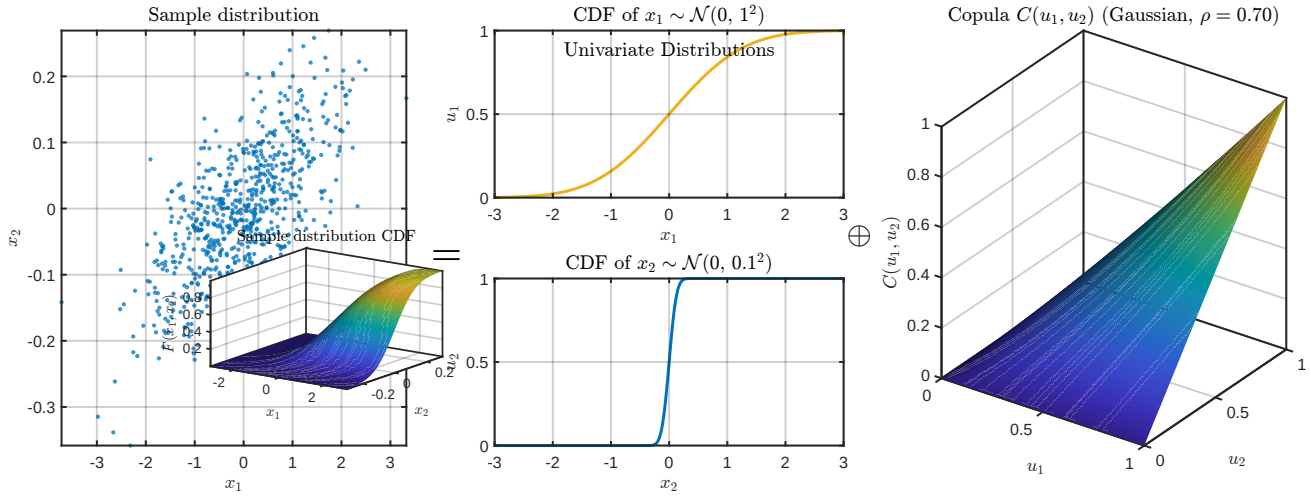

Figure 1: Schematic of a copula. *Left:* empirical sample (with its joint CDF inset). *Middle:* univariate CDFs F_1 and F_2 (mapping $x_j \mapsto u_j = F_j(x_j)$). *Right:* the copula surface $C(u_1, u_2)$, i.e., the joint CDF on $[0,1]^2$ with uniform marginals. The equality sign indicates Sklar's factorization $F(x_1, x_2) = C(F_1(x_1), F_2(x_2))$; the $\oplus$ highlights that C combines the marginal information into a dependence structure.

Biases b may be drawn independently from a simple one-dimensional law.

Design objective. Among samplers satisfying the dependence-matching and marginal constraints, choose $\mathcal{S}$ to improve downstream risk *without* any iterative tuning of W:

$$\min_{\mathcal{S}} \mathbb{E}_{W\sim\mathcal{S}(X)}\, \mathbb{E}_{(x,y)\sim\mathcal{D}}\big[\,\ell\big(h(x;W,b)\,\Theta^\star(W),\, y\big)\big]$$
$$\text{s.t. Copula}(w) \approx \widehat{\text{Copula}}(X), \quad w_j \sim G, \quad j = 1,\dots,d, \tag{7}$$

where $h(x;W,b) = \phi(xW+b) \in \mathbb{R}^h$ and $\mathcal{D}$ is the data distribution. In short, the problem is to replace i.i.d. frozen weights with *dependence-aware* frozen weights that align with the empirical copula of X, thereby addressing the core weakness of random initialization while preserving the hallmark efficiency of RdNNs.

4 Method

We introduce *Copula-Aligned Weight Initialization* (CAWI), a data-aware procedure that samples the frozen hidden weights in randomized neural networks using only the inputs X. CAWI preserves the classical RdNN pipeline (no backpropagation; closed-form output layer) while aligning the *dependence* among weight coordinates with that of the inputs. Concretely, CAWI (i) maps each feature to $[0,1]$ via its empirical cumulative distribution function to remove marginal scales, (ii) fits a multivariate copula on $[0,1]^d$ to summarize input dependence, and (iii) samples each weight column by drawing u from the fitted copula and transforming coordinates through a fixed marginal law G (e.g., $\mathcal{N}(0,1)$ or $\mathcal{U}[-1,1]$) via inverse CDFs. The training objective and closed-form solver remain unchanged; only the law used to sample (and then freeze) W is replaced. The detailed CAWI procedure and its steps are presented next. Algorithm 1 summarizes the procedure.

Step 1: Probability-Integral Transform of Features Given $X \in \mathbb{R}^{m\times d}$ with columns $X_{:j}$, we construct pseudo-observations

$$U_{ij} \;=\; \widehat{F}_j(X_{ij}) \in (0,1), \qquad U \in (0,1)^{m\times d},$$

where $\widehat{F}_j$ is the empirical cumulative distribution function (ECDF) of feature j, computed across samples (i.e., with j fixed and $i = 1,\dots,m$). A common rank-based implementation sets $U_{ij} = \frac{\text{rank}(X_{ij})}{m+1}$, where ranks are assigned feature-wise across samples. This guarantees $U_{ij} \in (0,1)$ (never exactly 0 or 1) and thus avoids infinities when applying inverse CDFs in Step 3.

This transformation removes units and strictly monotone re-scalings, isolating cross-feature *dependence* in the joint distribution of U, consistent with standard empirical copula construction.

Step 2: Fit a Multivariate Copula on U Let $\mathcal{C} = \{\,C(\cdot;\eta) : \eta \in \Xi\,\}$ be a parametric family of d-variate copulas on $[0,1]^d$, with parameter η in parameter space Ξ. Using the pseudo-observations U, we estimate $\widehat{\eta}$ to obtain a fitted copula $\widehat{C}(\cdot) = C(\cdot;\widehat{\eta})$, which summarizes the empirical dependence of the inputs and has uniform marginals by construction. Estimation is rank-based: one may maximize the copula pseudo-likelihood, solve method-of-moments equations

(e.g., matching Kendall's τ), or employ composite likelihoods over pairs when d is large. This fit is performed *once* per training split and is agnostic to the downstream task. The procedure is copula-agnostic: any d-variate copula family for which an estimator and a sampler are available (elliptical or Archimedean) can be plugged into $\mathcal{C}$. Concrete instantiations used in our work are detailed in 4.1.

Step 3: Sample Dependence-Aligned Weight Columns Choose a continuous one-dimensional marginal law G with a strictly increasing quantile G^{-1} for individual weight entries (e.g., $\mathcal{N}(0,1)$ or $\mathcal{U}[-1,1]$). For each hidden unit $t = 1, \ldots, h$:

1. draw $u^{(t)} \sim \widehat{C}$ in $[0,1]^d$;
2. set $w^{(t)} \leftarrow G^{-1}\big(u^{(t)}\big)$ coordinatewise, i.e., $w_j^{(t)} \leftarrow G^{-1}(u_j^{(t)})$ for $j = 1, \ldots, d$;
3. assign the t^{th} column of W as $W_{:t} \leftarrow w^{(t)}$ and draw a scalar bias b_t independently from a simple one-dimensional law.

Since G^{-1} is strictly increasing for continuous G, the *joint law of the coordinates of* $w^{(t)}$ has copula $\widehat{C}$ while its univariate marginals are G, thereby matching input dependence while retaining convenient weight marginals.

Step 4: Closed-Form Training (Unchanged) With the dependence-aligned weights W, we form the hidden matrix $H = \phi\big(XW + \mathbf{1}_m b^\top\big)$, and obtain the output layer by the standard ridge-regularized least squares

$$\Theta^\star \;=\; \arg\min_{\Theta} \; \|H\Theta - Y\|_F^2 + \frac{\lambda}{2}\|\Theta\|_F^2.$$

No iterative updates of W are introduced: CAWI modifies only the sampling law used to freeze the hidden weights; the training and inference pipeline remains identical to a classical RdNN.

Algorithm 1: CAWI: Copula–Aligned Weight Initialization for Randomized Neural Networks

Input : Data $X \in \mathbb{R}^{m\times d}$, width h, activation ϕ, ridge λ, weight marginal G, copula family $\mathcal{C} = \{C(\cdot;\eta) : \eta \in \Xi\}$.

Output : Frozen weights $W \in \mathbb{R}^{d\times h}$, bias vector $b \in \mathbb{R}^h$, output weights $\Theta^\star$.

```
// Step 1: Probability−integral transform (pseudo−observations)
for j ← 1 to d do
    compute ECDF F̂_j of column X_{:j}
    for i ← 1 to m do
        U_ij ← F̂_j(X_ij) ∈ (0,1)
// Step 2: Fit a d−variate copula on U (details in 4.1)
Estimate η̂ from U (rank–based); set Ĉ(·) ← C(·; η̂)
// Step 3: Sample dependence−aligned weight columns
for t ← 1 to h do
    draw u^(t) ~ Ĉ in [0,1]^d
    for j ← 1 to d do
        w_j^(t) ← G^{-1}(u_j^(t))
    set W_{:t} ← w^(t)
Sample each bias b_t independently from a simple one–dimensional law (e.g., uniform), for t = 1, ..., h
// Step 4: Closed−form output layer (unchanged)
Form hidden features H ← φ(XW + 1_m b^⊤)
Compute Θ*
```

4.1 Copula Families: Parameterization and Estimation

We instantiate Step 2 with five widely used d-variate copula families that together capture symmetric vs. asymmetric dependence and both light- and heavy-tailed co-movement: *Gaussian* (elliptical, tail-neutral), t (elliptical, heavy-tailed), *Clayton* (Archimedean, lower-tail dependent), *Gumbel* (Archimedean, upper-tail dependent), and *Frank* (Archimedean, tail-neutral, non-elliptical). This selection balances expressive coverage with tractable rank-based estimation and efficient sampling, enabling CAWI to adapt to diverse empirical dependence while keeping the training pipeline simple and stable.

Elliptical families (Gaussian, t) Both families are parameterized by a correlation matrix $R \in \mathbb{R}^{d\times d}$; the t copula additionally has degrees of freedom $\nu > 2$. We estimate R from pairwise Kendall's τ computed on U via the elliptical identity

$$\tau_{ij} \;=\; \tfrac{2}{\pi}\arcsin(R_{ij}) \Rightarrow \widehat{R}_{ij} \;=\; \sin\big(\tfrac{\pi}{2}\,\widehat{\tau}_{ij}\big)\text{(elementwise)},$$

followed by a nearest-correlation projection to enforce symmetry, unit diagonal, and positive (semi)definiteness (a vanishing ridge may be added if needed for numerical stability). For the t copula, we then profile a rank-based pseudo-likelihood over ν with R fixed to obtain $\widehat{\nu}$ (a one-dimensional search). The fitted copula $\widehat{C}$ yields draws $U \in (0,1)^{N\times d}$, which are

mapped to weights through the inverse marginals in Step 3.

Archimedean families (Clayton, Gumbel, Frank) These families are governed by a single scalar parameter θ that controls concordance and tail behavior. We employ robust, composite, rank-based estimation from pairwise Kendall's τ computed on U. Let $\widehat{\tau}_{ij}$ denote the sample Kendall's τ between coordinates i and j, and let $\bar{\tau}$ be their average over $i < j$. With this setup, we estimate a single θ per family as follows.

- **Clayton** ($\theta > 0$, lower-tail dependent). The τ–θ relation $\tau = \theta/(\theta + 2)$ yields
$$\widehat{\theta} \;=\; \frac{2\,\bar{\tau}}{1-\bar{\tau}}\,,$$
or, equivalently, one may minimize a composite method-of-moments objective $\sum_{i<j}\big(\frac{\theta}{\theta+2} - \widehat{\tau}_{ij}\big)^2$.
- **Gumbel** ($\theta \geq 1$, upper-tail dependent). With $\tau = 1 - \frac{1}{\theta}$,
$$\widehat{\theta} \;=\; \frac{1}{1-\bar{\tau}}\,,$$
or via the same composite fit $\sum_{i<j}\big(1 - \frac{1}{\theta} - \widehat{\tau}_{ij}\big)^2$.
- **Frank** ($\theta \in \mathbb{R} \setminus \{0\}$, tail-neutral, non-elliptical). Kendall's τ satisfies
$$\tau(\theta) \;=\; 1 - \frac{4}{\theta} + \frac{4}{\theta}\,D_1(\theta),$$
where D_1 is the Debye function of order 1. We obtain $\widehat{\theta}$ by numerically inverting $\tau(\theta)$ at $\bar{\tau}$ (monotone one-dimensional root-finding), or by composite pseudo-likelihood.

For $d > 2$ we adopt the standard exchangeable extension, i.e., a single global parameter $\widehat{\theta}$ shared across dimensions. This delivers stable, scalable d-variate fits and admits efficient generator-based sampling of $U \in (0,1)^{N\times d}$. As in Step 3, the sampled U is then mapped coordinatewise through the chosen marginal quantile G^{-1} to produce dependence-aligned weight columns.

4.2 Computational Complexity of CAWI

CAWI adds a one-time preprocessing stage (Steps 1-3) on top of the RdNN pipeline. Let m denote the number of samples, d the input dimensionality, and h the width (number of hidden units).

Step 1: Probability-integral transform. Computing the empirical CDF or rank transform for each of the d feature columns requires sorting m values per column, leading to a total cost of $O(d\,m\log m)$.

Step 2: Copula fitting. Using rank-based Kendall's τ for dependence estimation requires computing pairwise concordance statistics for all $d(d-1)/2 = O(d^2)$ feature pairs. A standard implementation of Kendall's τ costs $O(m^2)$ per pair, yielding an overall complexity of $O(d^2m^2)$. For elliptical copulas (Gaussian or t), an additional nearest-correlation or spectral projection step contributes $O(d^3)$. The t-copula further requires a one-dimensional search over degrees of freedom, with per-iteration cost $O(md^2)$, which remains negligible relative to the dominant $O(d^2m^2)$ term. For Archimedean copulas (Clayton, Gumbel, Frank), estimating a single scalar parameter from averaged Kendall's $\bar{\tau}$ incurs negligible cost beyond the $O(d^2m^2)$ computation of the pairwise τ-values.

Step 3: Sampling dependence-aligned weight columns. For elliptical copulas, sampling requires an initial factorization of the $d\times d$ correlation matrix, costing $O(d^3)$, followed by h matrix-vector multiplications, each costing $O(d^2)$, for a total of $O(d^3 + hd^2)$. For Archimedean copulas, sampling each column requires only $O(d)$ operations, yielding a total cost of $O(hd)$. In both cases, mapping the copula samples through the marginal quantile G^{-1} costs $O(hd)$, which is dominated by the previous terms.

Step 4: Closed-form RdNN training (unchanged). With dependence-aligned frozen weights, hidden features are formed as $H = \phi(XW + \mathbf{1}_m b^\top)$. The cost of forming XW is $O(mdh)$, and computing the ridge-regularized output layer using the normal equations costs $O(mh^2 + h^3)$. This is identical to classical RdNNs.

Summary. CAWI introduces a one-time preprocessing cost of $O\big(d\;m\log m + d^2m^2 + d^3 + hd^2\big)$, while the dominant runtime in practice remains the standard RdNN training step, $O(mdh + mh^2 + h^3)$. Thus, CAWI preserves the hallmark efficiency of RdNNs. The empirical training-time analysis reported in Section E of the Appendix further confirms this efficiency.

5 Empirical Results

We evaluate CAWI by integrating it into representative randomized architectures spanning *shallow*, *deep*, and *wide* settings: *(i) RVFL* (single hidden layer with direct links) Pao et al. (1994), *(ii) dRVFL* (deep RVFL) Shi et al. (2021), and *(iii) BLS* (Broad Learning System; a wide, block-structured random-feature architecture) Chen and Liu (2017). We compare the standard i.i.d. baseline against CAWI variants that differ only in the fitted copula used to sample columns of W:

- **i.i.d. baseline**: W is sampled independently from a Uniform distribution ($\mathcal{U}[-1, 1]$).
- **CAWI (elliptical)**: W is sampled using *Gaussian* and t copulas.
- **CAWI (Archimedean)**: W is sampled using *Clayton*, *Frank*, and *Gumbel* copulas.

The detailed experimental setup is provided in Section B of the Appendix.

5.1 Dataset Description

The evaluation is conducted on 83 benchmark datasets downloaded from the UCI repository Dua and Graff (2017), comprising 30 binary and 53 multiclass classification tasks. The number of classes ranges from 2 to 26. Sample sizes span from 24 to 58,000 instances, and the number of features ranges from 3 to 263, covering both small- and large-scale datasets as well as low- and high-dimensional regimes. Detailed dataset statistics are provided in Table 6 of the Appendix. To validate performance in real-world settings, we additionally evaluate CAWI on two biomedical datasets: the *BreaKHis* breast cancer dataset Spanhol et al. (2015) and *Schizophrenia ROI* dataset from the COBRE repository (`http://fcon_1000.projects.nitrc.org/indi/retro/cobre.html`). For BreaKHis, we use 1,240 histopathological image scans at 400× magnification, organized into benign and malignant classes. The benign subclasses are adenosis (AD, 106), fibroadenoma (FA, 237), phyllodes tumor (PT, 115), and tubular adenoma (TA, 130); the malignant subclasses are lobular carcinoma (LC, 137), papillary carcinoma (PC, 138), ductal carcinoma (DC, 208), and mucinous carcinoma (MC, 169). Feature extraction follows the procedure in Gautam et al. (2020). The Schizophrenia ROI dataset includes 72 schizophrenia subjects (ages 18–65; mean 38.1 ± 13.9 years) and 74 healthy controls (ages 18–65; mean 35.8 ± 11.5 years); ROI features are derived following Tanveer et al. (2022).

5.2 Performance Evaluation

In this subsection, we present empirical results on three groups of benchmarks: the BreaKHis dataset, the Schizophrenia ROI dataset, and a broader suite of 83 UCI benchmark datasets. For BreaKHis and Schizophrenia ROI, we evaluate RVFL, dRVFL, and BLS under the standard i.i.d. initialization and the proposed CAWI variants (Gaussian, t, Clayton, Frank, and Gumbel). Tables 1 and 2 report the BreaKHis results for RVFL and dRVFL, while Tables 3 and 4 present the corresponding results for the Schizophrenia ROI dataset. Results for BLS are provided in Tables 7–8 of the Appendix. For the broader benchmark evaluation, we report results on 83 UCI datasets (30 binary and 53 multiclass). The aggregate summary is presented in Table 5, while the complete per-dataset results are available in Tables 9–10 of the Appendix. All these results are reported using a fixed seed (`rng(42, 'twister')`); additional experiments with two independent seeds (7 and 123) are provided in Section C of the Appendix to assess seed sensitivity.

5.2.1 Result on BreaKHis Dataset

Table 1 shows that *every* pairwise task attains a higher test accuracy with a copula-aligned initialization than with the i.i.d. baseline (rightmost column > 0 for all 16 rows). Gains range from about $+1.07$ to $+5.60$ percentage points (e.g., TA vs. PC: $+5.60$, PT vs. LC: $+5.20$, PT vs. DC: $+4.64$), indicating that dependence-aware sampling of W provides a meaningful lift even in a shallow RVFL. Not every copula family dominates on every task; rather, the *best* variant shifts across pairs, consistent with heterogeneous dependence patterns in the features. Across the 16 pairs, the winning family varies: the t-copula is best on 5 tasks (AD vs. LC/MC; FA vs. LC; TA vs. LC; plus a three-way tie in FA vs. DC), *Gaussian* on 3 (e.g., AD vs. DC; FA vs. PC; PT vs. LC), *Clayton* on 2 (PT vs. DC; TA vs. DC), *Frank* on 4 (FA vs. MC; PT vs. MC/PC), and *Gumbel* on 2 (AD vs. PC; TA vs. PC). A three-way tie (Gaussian/t/Clayton) in FA vs. DC suggests that distinct copula families can approximate the relevant dependence equally well in some cases. Overall, every BreaKHis pair benefits from copula-aligned initialization (all improvements > 0), with gains up to $+5.60$ percentage points, reinforcing the central claim: injecting empirical dependence into the frozen projections consistently improves RVFL performance, while different copula families provide complementary advantages across tasks. Table 2 shows that copula-aligned initialization again improves over the i.i.d. baseline on *every* pair (rightmost column > 0 for all 16 rows), now with a wider spread of gains: from $+0.27$ to $+12.97$ percentage (e.g., PT vs. DC: $+12.97$, TA vs. PC: $+4.10$, PT vs. MC: $+3.87$). This indicates that dependence-aware sampling remains beneficial in deeper randomized stacks and can yield larger lifts when multiple frozen layers compound representational biases. The winning family varies by pair, with a clear pattern favoring Archimedean copulas: *Clayton* leads on **8** tasks, *Frank* on **4**, *Gumbel* on **2**, and the t-copula on **2**; *Gaussian* does not take a win in this setting. These outcomes are consistent with heterogeneous, often asymmetric dependence in the extracted features: Clayton/Frank/Gumbel (which capture different tail/asymmetry profiles) dominate, while elliptical models win when dependence is more symmetric.

Table 1: Results on the BreaKHis dataset for RVFL baseline and copula-based initializations (test accuracy, %). Best per row is **bold** and highlighted; the last column reports the improvement of the best method over the i.i.d. baseline.

Dataset	iid	Gaussian	*t*	Clayton	Frank	Gumbel	Improvement
AD vs. DC	71.0138	**73.2616**	72.3041	69.7440	70.0666	71.0292	**+2.2478**
AD vs. LC	62.1429	62.9507	**65.0425**	62.9932	63.7500	63.4099	**+2.8996**
AD vs. MC	63.6364	65.0909	**66.5455**	64.3636	65.0909	65.4545	**+2.9091**
AD vs. PC	60.2211	61.4796	62.2959	61.4371	62.3044	**62.7126**	**+2.4915**
FA vs. DC	63.5955	**65.3933**	**65.3933**	**65.3933**	64.2697	64.2697	**+1.7978**
FA vs. LC	65.7766	66.3027	**67.3622**	66.0324	66.0396	66.8432	**+1.5856**
FA vs. MC	59.8464	59.6025	60.3403	60.8461	**61.0720**	60.5932	**+1.2256**
FA vs. PC	68.0000	67.7333	**69.0667**	66.6667	67.7333	67.4667	**+1.0667**
PT vs. DC	67.1731	70.2933	71.1971	**71.8125**	69.9760	70.2933	**+4.6394**
PT vs. LC	57.0980	**62.2980**	60.2824	59.0510	59.0980	60.2667	**+5.2000**
PT vs. MC	61.9987	62.3120	61.9799	61.6228	**63.7343**	60.9398	**+1.7356**
PT vs. PC	57.3569	58.1098	59.6941	58.5098	**59.7176**	58.1098	**+2.3607**
TA vs. DC	65.3995	65.9877	66.5847	**67.1817**	66.8745	66.8832	**+1.7822**
TA vs. LC	62.1663	62.8931	**63.6897**	63.6338	62.5577	62.5646	**+1.5234**
TA vs. MC	57.1751	59.5311	59.8418	58.5254	**60.8701**	58.5537	**+3.6950**
TA vs. PC	56.3382	56.7296	58.5674	57.4284	59.7065	**61.9427**	**+5.6045**

Table 2: Results on the BreaKHis dataset for dRVFL baseline and copula-based initializations (test accuracy, %). Best per row is **bold** and highlighted; the last column reports the improvement of the best method over the i.i.d. baseline.

Dataset	iid	Gaussian	*t*	Clayton	Frank	Gumbel	Improvement
AD vs. DC	74.1987	71.0087	69.7542	66.2366	**75.2514**	66.2366	**+1.0527**
AD vs. LC	62.5595	61.7602	61.3350	**67.5170**	65.8503	61.3180	**+4.9575**
AD vs. MC	67.2727	63.6364	65.0909	67.7273	**68.9091**	67.0909	**+1.6364**
AD vs. PC	65.9949	56.5391	56.5391	56.9473	**68.9541**	56.9473	**+2.9592**
FA vs. DC	66.5169	64.7191	66.0674	**67.6338**	67.5577	67.5646	**+1.1169**
FA vs. LC	69.7802	67.9063	67.6505	68.1802	68.4505	**70.0541**	**+0.2739**
FA vs. MC	58.3740	58.3740	59.1117	58.3740	58.3740	**60.5781**	**+2.2041**
FA vs. PC	72.8000	70.2933	73.1971	**74.8125**	72.9760	73.2933	**+2.0125**
PT vs. DC	64.4135	72.4375	68.7260	64.4135	**77.3846**	64.4135	**+12.9711**
PT vs. LC	55.8980	57.4667	56.3137	**57.8824**	57.0980	56.2745	**+1.9844**
PT vs. MC	61.9862	60.2130	60.5576	**65.8521**	61.2657	60.5576	**+3.8659**
PT vs. PC	54.9725	54.5725	**57.7412**	55.3804	54.9725	56.1725	**+2.7687**
TA vs. DC	61.5496	61.5496	61.5496	**62.2137**	61.5496	61.5496	**+0.6641**
TA vs. LC	68.1831	63.6408	63.7107	**69.2802**	67.0510	65.9329	**+1.0971**
TA vs. MC	57.1751	56.8418	56.8418	**60.8531**	56.8418	56.8418	**+3.6780**
TA vs. PC	57.0860	58.6164	**61.1810**	57.8407	57.4563	57.0720	**+4.0950**

Table 3: Results on the Schizophrenia ROI datasets for RVFL baseline and copula-based initializations (test accuracy, %). Best per row is **bold** and highlighted; the last column reports the improvement of the best method over the i.i.d. baseline.

Dataset	iid	Gaussian	*t*	Clayton	Frank	Gumbel	Improvement
SZ–ROI GM+WM	60.3448	61.7701	63.0805	63.1034	62.4138	**63.1264**	**+2.7816**
SZ–ROI GM	62.4138	**63.7931**	62.4368	62.4598	61.7931	61.1264	**+1.3793**
SZ–ROI WM	61.0805	60.3448	61.0575	**62.3908**	61.0345	59.6782	**+1.3103**

Table 4: Results on the Schizophrenia ROI datasets for dRVFL baseline and copula-based initializations (test accuracy, %). Best per row is **bold** and highlighted; the last column reports the improvement of the best method over the i.i.d. baseline.

Dataset	iid	Gaussian	*t*	Clayton	Frank	Gumbel	Improvement
SZ–ROI GM+WM	71.2874	**74.7816**	72.6437	71.9770	71.9310	72.6437	**+3.4942**
SZ–ROI GM	68.5287	69.3103	70.5747	68.5287	**70.6897**	68.5747	**+2.1610**
SZ–ROI WM	63.7701	64.4828	67.1494	**67.2184**	65.8391	66.5287	**+3.4483**

5.2.2 Result on Schizophrenia Dataset

Across the three ROI configurations (GM+WM, GM, WM), CAWI improves upon the i.i.d. baseline in both architectures. In the RVFL setting (see Table 3), the best copula varies by ROI: *Gumbel* leads on GM+WM (+2.78), *Gaussian* on GM (+1.38), and *Clayton* on WM (+1.31). For dRVFL (see Table 4), the pattern shifts: *Gaussian* tops GM+WM (+3.49 pp), *Frank* on GM (+2.16), and *Clayton* again wins on WM (+3.45). Overall, these results echo the BreaKHis findings without redundancy: dependence-aware sampling of W consistently lifts accuracy, and the most effective copula family depends on the ROI subset and architecture—underscoring the value of modeling the *type* of dependence (symmetric vs. asymmetric; central vs. tail) rather than assuming independence.

5.3 Results on 83 UCI Datasets

On the broader benchmark suite, we evaluate RVFL on 83 UCI datasets. To provide a concise and transparent summary of the empirical findings, we report the aggregate performance of CAWI in Table 5. For each dataset, we compare the conventional i.i.d. initialization with the best-performing CAWI configuration (i.e., the copula family yielding the highest test accuracy for that dataset). This reflects the intended usage of CAWI as a flexible, copula-agnostic framework capable of adapting to dataset-specific dependency structures. Across all 83 datasets, the best CAWI variant outperforms the i.i.d. baseline on every dataset. The average test accuracy of the i.i.d. initialization is 78.08%, whereas the average accuracy of the best CAWI configuration reaches 79.35%, corresponding to a mean improvement of +1.27 percentage points. These results demonstrate that CAWI yields consistent improvements across both binary and multiclass problems, rather than isolated gains on a small subset of datasets.

Table 5: Aggregate performance summary across 83 UCI datasets.

Metric	Value
Mean accuracy (i.i.d.)	78.08%
Mean accuracy (Best CAWI)	79.35%
Mean improvement	+1.27%
Datasets where CAWI > i.i.d.	83 / 83

6 Conclusions

In this work, we address a foundational gap in randomized neural networks, namely that input-to-hidden weights are randomly initialized and then held fixed throughout training. We present CAWI, a plug-in initialization scheme that aligns the frozen weight columns with the empirical dependence structure of the inputs via a fitted copula, while leaving the classical RdNN pipeline—closed-form readout, no backpropagation—unchanged. By modifying only the sampling law for W, CAWI yields dependence-aware hidden projections with a computational footprint comparable to i.i.d. initialization. Building on this design, our empirical study shows consistent and often substantial gains across 83 UCI benchmarks and two biomedical datasets (BreaKHis and Schizophrenia ROI) using RVFL, dRVFL, and BLS architectures, all while preserving backpropagation-free training. Codes are provided at `https://github.com/mtanveer1/CAWI`.

A natural next step for future research is the development of automatic copula selection strategies. Rather than manually evaluating multiple copula families, a data-driven mechanism could identify the most suitable dependency structure for a given dataset and initialize the model accordingly. Such an adaptive selection procedure would further streamline CAWI while preserving its computational efficiency and enhancing its ability to capture dataset-specific dependence patterns.

Acknowledgement

Mushir Akhtar acknowledges the financial support received from the CSIR, New Delhi, under Fellowship Grant No. 09/1022(13849)/2022-EMR-I. Mohd. Arshad acknowledges the funding support provided by the ANRF, India, through the Core Research Grant (Grant No. CRG/2023/001230).

References

Mushir Akhtar, A Kumari, M Sajid, A Quadir, Mohd Arshad, P N Suganthan, and M Tanveer. Towards robust and inversion-free randomized neural networks: The XG-RVFL framework. *Pattern Recognition*, page 112711, 2025. URL 10.1016/j.patcog.2025.112711.

Arjun Ashok, Étienne Marcotte, Valentina Zantedeschi, Nicolas Chapados, and Alexandre Drouin. TACTis-2: Better, faster, simpler attentional copulas for multivariate time series. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=xtOydkE1Ku.

Weipeng Cao, Xizhao Wang, Zhong Ming, and Jinzhu Gao. A review on neural networks with random weights. *Neurocomputing*, 275:278–287, 2018.

Andrea Ceni. Random orthogonal additive filters: a solution to the vanishing/exploding gradient of deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

David Chapman and Parniyan Farvardin. Interpretable measurement of cnn deep feature density using copula and the generalized characteristic function. *arXiv preprint arXiv:2411.05183*, 2024.

Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. *Advances in Neural Information Processing Systems*, 32, 2019.

CL Philip Chen and Zhulin Liu. Broad learning system: An effective and efficient incremental learning system without the need for deep architecture. *IEEE Transactions on Neural Networks and Learning Systems*, 29(1):10–24, 2017.

Wuxing Chen, Kaixiang Yang, Zhiwen Yu, Feiping Nie, and CL Philip Chen. Adaptive broad network with graph-fuzzy embedding for imbalanced noise data. *IEEE Transactions on Fuzzy Systems*, 2025.

Dheeru Dua and Casey Graff. UCI machine learning repository. *URL http://archive. ics. uci. edu/ml*, 7 (1):62, 2017.

C. Gautam, P. K. Mishra, A. Tiwari, B. Richhariya, H. M. Pandey, S. Wang, M. Tanveer, and for the Alzheimer's Disease Neuroimaging Initiative. Minimum variance-embedded deep kernel regularized least squares method for one-class classification and its applications to biomedical data. *Neural Networks*, 123:191–216, 2020.

Ian Goodfellow. Deep learning, 2016.

Li Guo, Runze Li, and Bin Jiang. An ensemble broad learning scheme for semisupervised vehicle type classification. *IEEE Transactions on Neural Networks and Learning Systems*, 32(12):5287–5297, 2021.

Minghui Hu, Ruobin Gao, and Ponnuthurai Nagaratnam Suganthan. Self-distillation for randomized neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 35(11):16119–16128, 2024.

Wei Hu, Lechao Xiao, and Jeffrey Pennington. Provable benefit of orthogonal initialization in optimizing deep linear networks. In *The Eighth International Conference on Learning Representations*, 2020.

Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

Boris Igelnik and Yoh-Han Pao. Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE Transactions on Neural Networks*, 6(6):1320–1329, 1995.

Ajay Jaiswal, Peihao Wang, Tianlong Chen, Justin Rousseau, Ying Ding, and Zhangyang Wang. Old can be gold: Better gradient flow can make vanilla-gcns great again. In *Advances in Neural Information Processing Systems*, volume 35, pages 7561–7574, 2022.

Harry Joe. *Dependence modeling with copulas*. CRC press, 2014.

Agnan Kessy, Alex Lewin, and Korbinian Strimmer. Optimal whitening and decorrelation. *The American Statistician*, 72(4):309–314, 2018.

Ivano Lauriola, Alberto Lavelli, and Fabio Aiolli. An introduction to deep learning in natural language processing: Models, techniques, and tools. *Neurocomputing*, 470:443–456, 2022.

Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade*, pages 9–50. Springer, 2002.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Nunzio A Letizia, Nicola Novello, and Andrea M Tonello. Copula density neural estimation. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

Yicheng Liu, Jie Wen, Chengliang Liu, Xiaozhao Fang, Zuoyong Li, Yong Xu, and Zheng Zhang. Language-driven cross-modal classifier for zero-shot multi-label image recognition. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 32173–32183, 2024.

Bingjun Luo, Junjie Zhu, Tianyu Yang, Sicheng Zhao, Chao Hu, Xibin Zhao, and Yue Gao. Learning deep hierarchical features with spatial regularization for

one-class facial expression recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 6065–6073, 2023.

Meenal V Narkhede, Prashant P Bartakke, and Mukul S Sutaone. A review on weight initialization strategies for neural networks. *Artificial Intelligence Review*, 55(1):291–322, 2022.

Deanna Needell, Aaron A Nelson, Rayan Saab, Palina Salanevich, and Olov Schavemaker. Random vector functional link networks for function approximation on manifolds. *Frontiers in Applied Mathematics and Statistics*, 10:1284706, 2024.

Roger B Nelsen. *An introduction to copulas*. Springer, 2006.

Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita. A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 32(2):604–624, 2021. doi: 10.1109/TNNLS.2020.2979670.

Yoh-Han Pao, Gwang-Hoon Park, and Dejan J Sobajic. Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing*, 6(2):163–180, 1994.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, 2013.

Chang-E Ren, Siyao Cheng, and Zehua Xuan. Foatlbl: Federated online active transfer learning via broad learning system. *Neurocomputing*, page 131363, 2025.

M. Sajid, A. K. Malik, and M. Tanveer. Intuitionistic fuzzy broad learning system: Enhancing robustness against noise and outliers. *IEEE Transactions on Fuzzy Systems*, 32(8):4460–4469, 2024a. doi: 10.1109/TFUZZ.2024.3400898.

M. Sajid, A. K. Malik, M. Tanveer, and Ponnuthurai N. Suganthan. Neuro-fuzzy random vector functional link neural network for classification and regression problems. *IEEE Transactions on Fuzzy Systems*, 32 (5):2738–2749, 2024b.

M Sajid, A Quadir, M Tanveer, and for the Alzheimer's Disease Neuroimaging Initiative. GB-RVFL: Fusion of randomized neural network and granular ball computing. *Pattern Recognition*, 159:111142, 2025a.

M. Sajid, M. Tanveer, and Ponnuthurai N. Suganthan. Ensemble deep random vector functional link neural network based on fuzzy inference system. *IEEE Transactions on Fuzzy Systems*, 33(1):479–490, 2025b.

Qiushi Shi, Rakesh Katuwal, Ponnuthurai N Suganthan, and Muhammad Tanveer. Random vector functional link neural network based ensemble deep learning. *Pattern Recognition*, 117:107978, 2021.

Giora Simchoni and Saharon Rosset. Flexible copula-based mixed models in deep learning: A scalable approach to arbitrary marginals. In *International Conference on Artificial Intelligence and Statistics*, pages 91–99. PMLR, 2025.

M Sklar. Fonctions de répartition à n dimensions et leurs marges. In *Annales de l'ISUP*, volume 8, pages 229–231, 1959.

Fabio A Spanhol, Luiz S Oliveira, Caroline Petitjean, and Laurent Heutte. A dataset for breast cancer histopathological image classification. *IEEE Transactions on Biomedical Engineering*, 63(7):1455–1462, 2015.

Ponnuthurai N Suganthan and Rakesh Katuwal. On the origins of randomization-based feedforward neural networks. *Applied Soft Computing*, 105:107239, 2021.

Sophia Sun and Rose Yu. Copula conformal prediction for multi-step time series forecasting. *arXiv preprint arXiv:2212.03281*, 2022.

Mohammad Tanveer, M A Ganaie, A Bhattacharjee, and C T Lin. Intuitionistic fuzzy weighted least squares twin SVMs. *IEEE Transactions on Cybernetics*, 53(7):4400–4409, 2022.

Jinyue Tian, Hui Xue, Yanfang Xue, and Pengfei Fang. Copula-nested spectral kernel network. In *Forty-first International Conference on Machine Learning*, 2024.

Rishabh Tiwari, Arnav Chavan, Deepak Gupta, Gowreesh Mago, Animesh Gupta, Akash Gupta, Suraj Sharan, Yukun Yang, Shanwei Zhao, Shihao Wang, et al. Rcv2023 challenges: Benchmarking model training and inference for resource-constrained deep learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1534–1543, 2023.

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

Cheryl Sze Yin Wong, Guo Yang, Arulmurugan Ambikapathi, and Ramasamy Savitha. Online continual learning using enhanced random vector functional link networks. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1905–1909. IEEE, 2022.

Fan Yin, Jayanth Srinivasa, and Kai-Wei Chang. Characterizing truthfulness in large language model generations with local intrinsic dimension. In *Proceedings*

of the 41st International Conference on Machine Learning, volume 235, pages 57069–57084, 2024.

Le Zhang and Ponnuthurai N Suganthan. A survey of randomized algorithms for training neural networks. *Information Sciences*, 364:146–155, 2016.

Weijia Zhang, Chun Kai Ling, and Xuanhui Zhang. Deep copula-based survival analysis for dependent censoring with identifiability guarantees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20613–20621, 2024.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Appendix

A Architectural and Mathematical Formulation of Standard RdNN Models

A.1 Formulation and Architecture of RVFL and ELM Pao et al. (1994); Huang et al. (2006)

RVFL and ELM are single-layer feed-forward RdNNs that rely on hidden layer transformations and closed-form solutions for output weight computation. While their architectures share significant similarities, the primary distinction lies in the presence of direct input-to-output connections in RVFL, which are absent in ELM. Their common formulation is described as follows:
Hidden Layer: The hidden layer transforms the input data X using a random weight matrix $W \in \mathbb{R}^{d\times h}$ and bias $B \in \mathbb{R}^{m\times h}$:

$$H = \phi(XW + B), \tag{8}$$

where $\phi(\cdot)$ is the non-linear function and h represents the number of nodes in the hidden layer.
Output Layer: In RVFL, the final output combines the hidden layer outputs and the original inputs to form an augmented representation:

$$A = [X \mid H], \tag{9}$$

whereas in ELM, the output is directly derived from the hidden layer outputs:

$$A = H. \tag{10}$$

The output weight matrix $\Theta \in \mathbb{R}^{h\times n_{\text{class}}}$ is computed in both architectures using a closed-form solution:

$$\Theta = (A^\top A + \lambda I)^{-1} A^\top Y, \tag{11}$$

where λ is a regularization parameter.

Key Difference: RVFL incorporates direct input-to-output connections ($Z = [X \mid H]$), which preserve input information and enhance robustness. In contrast, ELM omits this feature ($Z = H$), resulting in a simpler architecture that trades off some robustness for faster training.

A.2 Formulation and Architecture of BLS Chen and Liu (2017)

BLS represents a significant advancement in RdNNs by utilizing a flat architecture instead of the deep hierarchical structures seen in traditional deep learning. The architecture of BLS consists of three primary components: the feature layer, the enhancement layer, and the output layer. The feature layer is responsible for extracting meaningful features from the input data by generating multiple groups of feature nodes through random projection and non-linear transformations. On the other hand, the enhancement layer enriches the feature representations by applying additional non-linear transformations, thereby constructing enhancement nodes that provide diversity and robustness. The outputs from both layers are concatenated to form a final matrix, which is used to compute the output weights via a closed-form solution. Its architecture can be described as follows:
Feature Layer: For the input matrix X, the feature layer generates q windows of feature nodes. Each window f_i contains p nodes, which are computed as:

$$Z_{f_i} = \phi(XW_{f_i} + B_{f_i}), \tag{12}$$

where $W_{f_i} \in \mathbb{R}^{d\times p}$ is a random weight matrix, $B_{f_i} \in \mathbb{R}^{m\times p}$ is a bias matrix, $\phi(\cdot)$ is a non-linear function, and $Z_{f_i} \in \mathbb{R}^{m\times p}$ represents the output of the i^{th} feature node window. The outputs of all feature windows are concatenated to form the overall feature layer output:

$$Z = [Z_{f_1}, Z_{f_2}, \ldots, Z_{f_q}], \tag{13}$$

where $Z \in \mathbb{R}^{m \times pq}$. The feature layer output Z serves as the input to the enhancement layer.
Enhancement Layer: The enhancement layer enriches the feature representation by applying additional random transformations and a non-linear function to the concatenated feature nodes Z. Each window (e_j) of enhancement nodes (there are s windows, each with r nodes) is computed as:

$$E_{e_j} = \psi(ZW_{e_j} + B_{e_j}), \tag{14}$$

where $Z \in \mathbb{R}^{m \times pq}$ is the output from the feature layer, $W_{e_j} \in \mathbb{R}^{pq \times r}$ is a random weight matrix, $B_{e_j} \in \mathbb{R}^{m \times r}$ is a bias matrix, $\psi(\cdot)$ is a non-linear function, and $E_{e_j} \in \mathbb{R}^{m \times r}$ is the output of the j^{th} enhancement node window. The outputs of all enhancement windows are concatenated to form the overall enhancement layer output:

$$E = [E_{e_1}, E_{e_2}, \ldots, E_{e_s}], \tag{15}$$

where $E \in \mathbb{R}^{m \times rs}$.
Output Layer: The final representation matrix A, which combines the outputs of the feature and enhancement layers, is given by:

$$A = [Z \mid E], \tag{16}$$

where $A \in \mathbb{R}^{m \times (pq+rs)}$.

The output weights $\Theta \in \mathbb{R}^{(pq+rs) \times n_{\text{class}}}$ are then learned using a closed-form solution:

$$\Theta = (A^\top A + \lambda I)^{-1} A^\top Y, \tag{17}$$

where λ is a regularization parameter.
Incremental Learning: One of BLS's unique features is its ability to incrementally update the network by adding new feature mapping nodes or enhancement nodes without retraining the entire model. This makes BLS computationally efficient and adaptable to streaming or evolving data.

A.3 Formulation and Architecture of deep RVFL Shi et al. (2021)

The deep random vector functional link network (dRVFL) extends the standard RVFL architecture by incorporating multiple hidden layers, thereby enhancing its representational capacity while preserving the hallmark efficiency of closed-form training. Unlike conventional deep neural networks, dRVFL stacks several RVFL-like layers where only the final layer's output weights are trained via least squares, and all intermediate transformations are computed using fixed, randomly initialized weights. The architecture can be described as follows:

Layer-wise Random Transformations: Let the input matrix be $X \in \mathbb{R}^{m \times d}$. The l^{th} hidden layer transformation is defined as:

$$H^{(l)} = \phi(H^{(l-1)} W^{(l)} + B^{(l)}), \quad l = 1, 2, \ldots, L, \tag{18}$$

where $H^{(0)} = X$, $W^{(l)} \in \mathbb{R}^{h_{l-1} \times h_l}$ is the randomly initialized weight matrix for layer l, $B^{(l)} \in \mathbb{R}^{m \times h_l}$ is the corresponding bias matrix, and $\phi(\cdot)$ is a nonlinear activation function applied elementwise.

Augmented Feature Representation: The final feature representation A is constructed by concatenating the original input and the outputs of all hidden layers:

$$A = [X \mid H^{(1)} \mid H^{(2)} \mid \ldots \mid H^{(L)}] \in \mathbb{R}^{m \times a} \tag{19}$$

where $a = d + \sum_{l=1}^{L} h_l$ is the total feature dimensionality. This dense aggregation of features enables dRVFL to capture increasingly abstract representations at deeper layers while retaining the original input.

Output Layer: The output weights $\Theta \in \mathbb{R}^{a \times n_{\text{class}}}$ are obtained using the same closed-form solution as in RVFL:

$$\Theta = (A^\top A + \lambda I)^{-1} A^\top Y, \tag{20}$$

where $\lambda \geq 0$ is the regularization parameter.

Table 6: Description of binary and multiclass datasets used in the experiments. The table lists the dataset name, number of samples, number of features, and number of classes.

Dataset	Number of Samples	Number of Features	Number of Classes	Dataset	Number of Samples	Number of Features	Number of Classes
Multiclass datasets							
abalone	4177	8	3	synthetic_control	600	60	6
annealing	898	31	5	thyroid	7200	21	3
arrhythmia	452	262	13	vertebral_column_3clases	310	6	3
balance_scale	625	4	3	wall_following	5456	24	4
cardiotocography_10clases	2126	21	10	waveform	5000	21	3
cardiotocography_3clases	2126	21	3	waveform_noise	5000	40	3
conn_bench_vowel_deterding	990	11	11	wine	178	13	3
contrac	1473	9	3	wine_quality_red	1599	11	6
dermatology	366	34	6	wine_quality_white	4898	11	7
ecoli	336	7	8	yeast	1484	8	10
energy_y1	768	8	3	zoo	101	16	7
energy_y2	768	8	3	**Binary datasets**			
flags	194	28	8	bank	4521	16	2
glass	214	9	6	blood	748	4	2
heart_cleveland	303	13	5	breast_cancer	286	9	2
heart_switzerland	123	12	5	breast_cancer_wisc	699	9	2
image_segmentation	2310	18	7	breast_cancer_wisc_diag	569	30	2
iris	150	4	3	breast_cancer_wisc_prog	198	33	2
led_display	1000	7	10	chess_krvkp	3196	36	2
lenses	24	4	3	congressional_voting	435	16	2
letter	20000	16	26	conn_bench_sonar_mines_rocks	208	60	2
low_res_spect	531	100	9	credit_approval	690	15	2
lymphography	148	18	4	cylinder_bands	512	35	2
molec_biol_splice	3190	60	3	echocardiogram	131	10	2
nursery	12960	8	5	haberman_survival	306	3	2
oocytes_merluccius_states_2f	1022	25	3	hepatitis	155	19	2
oocytes_trisopterus_states_5b	912	32	3	horse_colic	368	25	2
optical	5620	62	10	ilpd_indian_liver	583	9	2
page_blocks	5473	10	5	ionosphere	351	33	2
pendigits	10992	16	10	monks_1	556	6	2
pittsburg_bridges_MATERIAL	106	7	3	monks_3	554	6	2
pittsburg_bridges_REL_L	103	7	3	oocytes_merluccius_nucleus_4d	1022	41	2
pittsburg_bridges_SPAN	92	7	3	oocytes_trisopterus_nucleus_2f	912	25	2
pittsburg_bridges_TYPE	105	7	6	pima	768	8	2
post_operative	90	8	3	planning	182	12	2
seeds	210	7	3	spect	265	22	2
semeion	1593	256	10	spectf	267	44	2
soybean	683	35	18	statlog_australian_credit	690	14	2
statlog_image	2310	18	7	statlog_german_credit	1000	24	2
statlog_landsat	6435	36	6	statlog_heart	270	13	2
statlog_shuttle	58000	9	7	tic_tac_toe	958	9	2
statlog_vehicle	846	18	4	titanic	2201	3	2

B Experimental Setup and Hyperparameter Setting

All the experiments are implemented using MATLAB R2023a and executed on a Windows 10 PC equipped with an Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz (4 cores, 8 logical processors) and 16 GB RAM. Each dataset is preprocessed by normalizing the input features to have zero mean and unit variance. The description of 83 datasets (binary and multiclass) used in the experiments are provided in Table 6. A 5-fold cross-validation procedure is employed to ensure reliable and unbiased evaluation. In each fold, the dataset is split into 80% training data and 20% testing data. For every combination of hyperparameters, the model is trained on the training data and evaluated on the testing data across all 5 folds. The testing accuracy is recorded for each fold. The final testing accuracy for each dataset is computed as the mean testing accuracy across the five folds, providing a robust estimate of the model's performance.

To eliminate any possibility of data leakage, all copula estimation steps are performed strictly within each training split. For every fold, the pseudo-observations and copula parameters are computed exclusively using the training features of that fold. The fitted copula is then used to sample the hidden-layer weights, which are subsequently evaluated on the disjoint test set. At no stage is information from the test portion used during copula fitting or weight initialization.

Hyperparameter tuning is performed using a grid search strategy to identify the optimal settings for each model. For each model, the regularization parameter (λ) is selected from $\{10^i \mid i = -6, -5, \ldots, 6\}$. For RVFL, the number of hidden nodes (h) varies from $[3 : 20 : 203]$, and seven activation functions (Sigmoid (1), Sine (2), Tribas (3), Radbas (4), Tansig (5), ReLU (6), and SELU (7)) are evaluated. For BLS, the number of feature windows (q), the number of feature nodes in each window (p), and the number of enhancement nodes (r) are set as per Sajid et al. (2024a), with Tansig as the activation function. For dRVFL, we adopt the same hyperparameter settings as provided in Shi et al. (2021) and evaluate three activation functions: Sigmoid (1), ReLU (2), and SELU (3).

Table 7: Results on the BreaKHis dataset for BLS baseline and copula-based initializations (test accuracy, %). Best per row is **bold** and highlighted; the last column reports the improvement of the best method over the i.i.d. baseline.

Dataset	iid	Gaussian	*t*	Clayton	Frank	Gumbel	Improvement
AD vs. DC	70.7066	71.0394	70.8966	69.7440	**72.3041**	71.8292	**+1.5975**
AD vs. LC	63.1429	62.9507	**65.0425**	62.9932	63.7500	63.4099	**+1.8996**
AD vs. MC	63.8396	65.8909	**68.5455**	65.3636	66.0909	66.4545	**+4.7059**
AD vs. PC	61.2956	61.8296	62.2959	62.4371	63.3044	**64.2126**	**+2.9170**
FA vs. DC	64.1055	**65.3933**	**65.3933**	**65.3933**	64.2697	64.2697	**+1.2878**
FA vs. LC	67.8991	68.7099	**68.7135**	67.6468	68.1802	68.7099	**+0.8144**
FA vs. MC	60.8371	62.3156	61.3403	**63.8461**	61.0720	60.5932	**+3.0090**
FA vs. PC	68.5746	67.9333	**69.8667**	66.8667	67.9233	67.8667	**+1.2921**
PT vs. DC	68.1731	70.2933	71.1971	**71.8125**	69.9760	70.2933	**+3.6394**
PT vs. LC	57.5980	**62.2980**	60.2824	59.0510	59.0980	60.2667	**+4.7000**
PT vs. MC	62.5987	62.3120	61.9799	61.6228	**64.5243**	60.9398	**+1.9256**
PT vs. PC	58.1069	58.1098	59.6941	58.5098	**60.0176**	58.1098	**+1.9107**
TA vs. DC	62.9939	64.7937	**66.2818**	65.1141	65.4214	65.7024	**+3.2879**
TA vs. LC	65.9189	66.7086	68.1971	67.3934	67.4074	**68.9099**	**+2.9910**
TA vs. MC	60.1921	61.8814	**63.5593**	63.2260	**63.5593**	61.5254	**+3.3672**
TA vs. PC	61.1740	60.8036	**62.2851**	60.4472	60.0978	61.5933	**+1.1111**

Table 8: Results on the Schizophrenia ROI datasets for BLS baseline and copula-based initializations (test accuracy, %). Best per row is **bold** and highlighted; the last column reports the improvement of the best method over the i.i.d. baseline.

Dataset	iid	Gaussian	*t*	Clayton	Frank	Gumbel	Improvement
SZ–ROI GM+WM	71.9770	72.6897	74.0000	74.0690	74.0230	**74.7126**	**+2.7356**
SZ–ROI GM	70.6667	**73.2414**	71.9540	71.9080	69.9540	72.6207	**+2.5747**
SZ–ROI WM	67.9080	67.2184	**69.2184**	67.9080	67.9080	68.5287	**+1.3104**

C Multi-Seed Stability Analysis

To assess the robustness of CAWI with respect to randomness, we additionally evaluate the RVFL model under multiple independent random seeds. While all primary experiments were conducted using a fixed seed (`rng(42, 'twister')`), we re-run the experiments using two additional seeds (7 and 123) for both the i.i.d. baseline and all copula-based variants.

The detailed results for the BreaKHis dataset under Seed 7 and Seed 123 are reported in Tables 11 and 12, respectively. Similarly, the corresponding results for the Schizophrenia ROI dataset are presented in Tables 13 and 14.

Across both datasets and independent seeds, CAWI consistently outperforms the conventional i.i.d. initialization. Importantly, the relative improvements remain stable in magnitude and direction, indicating that the gains are not attributable to a particular random realization.

Overall, the multi-seed evaluation confirms that CAWI provides stable and reproducible performance improvements across independent random initializations, reinforcing its robustness as a dependency-aware weight initialization strategy.

D Statistical Significance Across Datasets

To assess whether the performance improvements of CAWI over the i.i.d. initialization are statistically significant across datasets, we conduct a non-parametric Wilcoxon signed-rank test over the 83 UCI benchmark datasets.

Table 9: Results on benchmark binary datasets for RVFL baseline and copula-based initializations (test accuracy, %). Best per row is **bold** and highlighted; the last column reports the improvement of the best method over the i.i.d. baseline.

Dataset	iid	Gaussian	t	Clayton	Frank	Gumbel	Improvement
bank	89.6705	89.8032	89.7809	89.6705	**89.8254**	89.7146	**+0.1549**
blood	76.7714	77.0389	**77.5732**	**77.5732**	77.4398	77.4398	**+0.8018**
breast_cancer	70.1754	70.1754	70.1754	70.1754	70.1754	**70.5263**	**+0.3509**
breast_cancer_wisc	88.5612	88.7040	**89.4193**	88.7040	88.9887	89.2775	**+0.8581**
breast_cancer_wisc_diag	94.7260	94.9014	**95.4308**	95.2554	95.0753	95.2523	**+0.7048**
breast_cancer_wisc_prog	81.8462	82.8462	**83.4231**	82.3718	82.8974	81.8846	**+1.5769**
chess_krvkp	81.4485	82.4172	82.3546	82.4174	**82.8535**	82.1356	**+1.4050**
congressional_voting	63.9080	63.6782	**64.1379**	63.6782	63.6782	63.6782	**+0.2299**
conn_bench_sonar_mines_rocks	61.9861	65.4123	**66.7944**	62.5900	64.4948	64.3786	**+4.8083**
credit_approval	85.2174	**85.7971**	85.6522	**85.7971**	85.6522	85.6522	**+0.5797**
cylinder_bands	69.7392	69.7335	**70.1085**	69.1376	68.9587	69.7373	**+0.3693**
echocardiogram	85.4416	85.4416	**86.2108**	85.4416	85.4416	85.4416	**+0.7692**
haberman_survival	74.1460	73.4902	**74.8017**	74.4738	74.4738	74.1460	**+0.6557**
hepatitis	85.1613	86.4516	86.4516	85.8065	85.8065	**87.0968**	**+1.9355**
horse_colic	85.8830	86.1570	**86.9604**	86.4198	86.4198	86.1570	**+1.0774**
ilpd_indian_liver	71.7006	72.7306	73.0725	72.3902	**73.5883**	72.7336	**+1.8877**
ionosphere	89.7626	90.0443	**91.1871**	90.0523	90.6117	90.6237	**+1.4245**
monks_1	82.5338	84.8745	**84.8858**	84.1506	84.5206	83.9704	**+2.3520**
monks_3	91.3415	92.0606	92.6011	92.2424	92.0573	**92.6044**	**+1.2629**
oocytes_merluccius_nucleus_4d	82.9775	83.1703	**83.2669**	82.8771	82.8780	83.0693	**+0.2894**
oocytes_trisopterus_nucleus_2f	79.1647	80.5861	80.2600	**81.1355**	80.0342	80.1417	**+1.9708**
pima	73.7043	73.9649	**74.2212**	73.9649	73.1856	73.7051	**+0.5169**
planning	71.3814	71.9369	71.9369	**72.4625**	71.9520	71.9369	**+1.0811**
spect	67.1698	68.3019	**69.8113**	69.0566	68.3019	68.6792	**+2.6415**
spectf	79.3431	79.3501	79.7205	79.7135	**79.7275**	79.3431	**+0.3844**
statlog_australian_credit	68.5507	68.5507	68.9855	**69.2754**	68.9855	68.9855	**+0.7247**
statlog_german_credit	76.8000	77.7000	77.7000	**78.0000**	77.5000	77.6000	**+1.2000**
statlog_heart	80.7407	81.4815	**82.9630**	81.8519	82.5926	81.1111	**+2.2223**
tic_tac_toe	89.9700	90.8077	**91.1229**	90.5923	89.9700	90.3878	**+1.1529**
titanic	77.9168	77.9168	**78.1901**	77.9168	77.9168	77.9168	**+0.2733**

For each dataset $i \in \{1, \ldots, 83\}$, let A_i^{iid} denote the test accuracy obtained using the conventional i.i.d. initialization, and let A_i^{CAWI} denote the test accuracy achieved by the best-performing CAWI configuration on the same dataset. We use the best CAWI variant per dataset because CAWI is designed as a flexible, copula-agnostic framework: different datasets exhibit different dependency structures, and therefore different copula families may be most appropriate for different tasks. This comparison reflects the intended usage of CAWI, where the copula family is selected to best capture the empirical dependence of each dataset.

We then construct paired observations

$$\left(A_i^{\text{iid}},\, A_i^{\text{CAWI}}\right), \quad i = 1, \ldots, 83,$$

and apply the Wilcoxon signed-rank test to the differences

$$D_i = A_i^{\text{CAWI}} - A_i^{\text{iid}}.$$

The resulting Wilcoxon statistic is $W = 3403$, while the maximum possible statistic for 83 paired observations is 3486. The corresponding p-value satisfies $p < 10^{-6}$, indicating extremely strong statistical evidence against the null hypothesis of equal median performance.

These results demonstrate that the observed improvements of CAWI over the i.i.d. initialization are not attributable to random fluctuations on individual datasets, but instead reflect a consistent and statistically significant advantage across diverse benchmark problems.

Table 10: Results on benchmark multiclass datasets for RVFL baseline and copula-based initializations (test accuracy, %). Best per row is **bold** and highlighted; the last column reports the improvement of the best method over the i.i.d. baseline.

Dataset	iid	Gaussian	*t*	Clayton	Frank	Gumbel	Improvement
abalone	63.5141	63.6813	63.8735	63.8728	63.9687	**64.0169**	**+0.5028**
annealing	89.5202	90.3048	**90.4115**	89.8603	90.3048	90.1906	**+0.8913**
arrhythmia	70.8010	71.0256	71.0183	70.8010	**71.2405**	**71.2405**	**+0.4395**
balance_scale	98.5600	98.5600	**98.7200**	**98.7200**	98.5600	**98.7200**	**+0.1600**
cardiotocography_10clases	**70.7455**	69.9461	70.6033	70.3695	70.2276	70.3207	**+0.0000**
cardiotocography_3clases	85.9859	86.3148	**86.6909**	86.5030	86.5023	86.2684	**+0.7050**
conn_bench_vowel_deterding	95.8586	96.1616	96.1616	**96.4646**	96.1616	96.1616	**+0.6060**
contrac	41.0681	**42.7001**	42.0883	41.4034	41.6765	42.0830	**+1.6320**
dermatology	97.2677	97.8119	**98.0859**	97.8119	97.8119	98.0822	**+0.8182**
ecoli	60.6234	61.2160	61.2116	61.2072	61.2160	**61.5057**	**+0.8823**
energy_y1	88.5307	89.1936	**89.9677**	88.6682	89.1945	89.0578	**+1.4370**
energy_y2	90.1002	90.7538	**91.6645**	91.2783	91.0126	91.0160	**+1.5643**
flags	53.6437	55.1822	55.2227	53.6302	**56.2078**	53.6437	**+2.5641**
glass	38.1617	39.0808	40.9413	**42.3367**	39.5570	40.0221	**+4.1750**
heart_cleveland	59.3661	60.0546	**60.7049**	60.3661	60.6940	59.7322	**+1.3388**
heart_switzerland	47.3000	47.9333	49.7000	48.0333	**49.7667**	47.3333	**+2.4667**
image_segmentation	88.1385	88.4848	**88.5714**	88.2251	**88.5714**	88.3117	**+0.4329**
iris	75.3333	**78.6667**	76.6667	76.0000	78.0000	77.3333	**+3.3334**
led_display	72.9000	73.2000	**73.7000**	73.3000	73.6000	73.3000	**+0.8000**
lenses	92.0000	**96.0000**	**96.0000**	**96.0000**	**96.0000**	**96.0000**	**+4.0000**
letter	80.8500	80.9050	80.9750	81.0350	80.9650	**81.1450**	**+0.2950**
low_res_spect	88.5117	88.5117	**89.4534**	89.0778	88.8891	88.8891	**+0.9417**
lymphography	86.4138	**88.5057**	87.7931	87.1034	87.7931	87.8161	**+2.0919**
molec_biol_splice	53.5110	54.9216	**56.3009**	53.8558	53.9812	54.4514	**+2.7899**
nursery	71.2114	70.8951	71.7978	70.9877	71.7747	**72.9784**	**+1.7670**
oocytes_merluccius_states_2f	91.5782	91.9699	**92.3649**	92.0679	91.9699	91.9694	**+0.7867**
oocytes_trisopterus_states_5b	87.9427	89.1491	**89.4806**	89.1509	88.9227	88.9275	**+1.5379**
optical	96.9929	97.2064	**97.2242**	97.1530	97.1886	97.1886	**+0.2313**
page_blocks	95.3771	95.7242	95.8157	95.5597	95.5781	**95.5963**	**+0.4386**
pendigits	98.5353	98.5990	98.6899	98.6626	98.6536	**98.7081**	**+0.1728**
pittsburg_bridges_MATERIAL	78.7013	76.8831	78.6147	76.8831	**80.4762**	78.7013	**+1.7749**
pittsburg_bridges_REL_L	63.1429	63.2381	68.8571	65.2381	66.2381	**66.8095**	**+5.7142**
pittsburg_bridges_SPAN	63.0409	64.2690	**67.4854**	64.2105	64.2105	63.2749	**+4.4445**
pittsburg_bridges_TYPE	42.8571	42.8571	**46.6667**	42.8571	44.7619	42.8571	**+3.8096**
post_operative	71.1111	71.1111	71.1111	71.1111	71.1111	**72.2222**	**+1.1111**
seeds	90.0000	89.5238	**90.4762**	**90.4762**	89.5238	90.0000	**+0.4762**
semeion	87.8208	88.4488	**89.0751**	88.9505	88.6369	88.4484	**+1.2543**
soybean	90.4755	90.1889	**91.0702**	90.1793	90.0333	90.0333	**+0.5947**
statlog_image	95.4113	95.4545	**95.6277**	95.5844	95.5844	95.5844	**+0.2164**
statlog_landsat	82.1134	82.6107	82.5486	82.5175	82.3155	**82.3310**	**+0.4973**
statlog_shuttle	98.7017	98.7448	**98.7534**	98.7190	98.7276	98.7224	**+0.0517**
statlog_vehicle	82.1511	82.5040	82.8597	**82.8611**	82.2694	**82.8611**	**+0.7100**
synthetic_control	54.0000	54.8333	55.1667	55.0000	**55.5000**	55.3333	**+1.5000**
thyroid	95.7639	95.7361	95.9028	95.7917	95.8056	**95.9444**	**+0.1805**
vertebral_column_3clases	65.1613	66.1290	66.4516	**66.7742**	65.8065	66.1290	**+1.6129**
wall_following	77.6955	78.2640	78.1717	78.0253	77.8052	**78.8133**	**+1.1178**
waveform	86.8800	86.9200	87.0000	87.0000	**87.0800**	86.8400	**+0.2000**
waveform_noise	86.3400	86.4200	86.5000	86.4800	86.3800	86.4200	**+0.1600**
wine	96.6349	96.6508	**97.2063**	96.6508	96.6508	96.6508	**+0.5714**
wine_quality_red	59.6638	60.0380	**60.9122**	60.6025	60.6023	60.6019	**+1.2484**
wine_quality_white	52.7576	52.8187	53.0842	52.9617	**53.1046**	52.9615	**+0.3470**
yeast	57.1424	57.7496	57.6169	**57.8840**	57.6815	57.3460	**+0.7416**
zoo	95.0000	**97.0000**	**97.0000**	95.0000	96.0000	96.0000	**+2.0000**

Table 11: BreaKHis dataset (Seed 7): Test accuracy (%). Best per row is **bold** and highlighted; the last column reports the improvement over the i.i.d. baseline.

Dataset	iid	Gaussian	*t*	Clayton	Frank	Gumbel	Improvement
AD vs. DC	70.6429	72.8934	**73.1562**	69.4815	70.3274	71.2948	**+2.5133**
AD vs. LC	62.4138	**65.2847**	64.7193	63.1562	63.9284	63.7451	**+2.8709**
AD vs. MC	63.9091	65.2727	64.9091	**66.7273**	65.2727	65.6364	**+2.8182**
AD vs. PC	60.5793	61.8926	62.1038	61.6704	**62.9381**	62.4859	**+2.3588**
FA vs. DC	63.2584	65.6742	64.9438	65.1124	**65.8876**	64.6067	**+2.6292**
FA vs. LC	66.1039	66.5783	66.9324	**67.5429**	66.3891	66.2108	**+1.4390**
FA vs. MC	59.6139	60.1847	**61.4893**	60.3284	60.8951	60.2556	**+1.8754**
FA vs. PC	68.1733	67.6267	68.8000	67.2000	**69.4667**	67.9467	**+1.2934**
PT vs. DC	67.5192	70.8365	**72.0479**	71.5841	69.8654	70.4712	**+4.5287**
PT vs. LC	57.2863	61.8745	60.7294	**62.4196**	59.6275	60.5490	**+5.1333**
PT vs. MC	61.7254	**63.8194**	62.4183	61.3562	63.2816	61.4729	**+2.0940**
PT vs. PC	57.0196	58.5882	59.3725	**60.1176**	59.4902	58.6667	**+3.0980**
TA vs. DC	65.7239	66.5104	**67.8529**	66.9274	66.4183	66.2104	**+2.1290**
TA vs. LC	62.0384	63.3827	63.1562	**63.9418**	62.9716	62.3194	**+1.9034**
TA vs. MC	57.4283	59.9274	**61.2638**	58.8951	60.5183	58.1029	**+3.8355**
TA vs. PC	56.5071	57.2849	58.9427	**61.8394**	59.3184	58.4716	**+5.3323**

E Empirical Training-Time Analysis

To complement the theoretical complexity discussion provided in the Section 4.2 of the main paper, we report an empirical evaluation of the training time incurred by CAWI under different copula families. Specifically, we measure the average training time (in seconds) of the RVFL model initialized using five copula families, Gaussian, t, Clayton, Frank, and Gumbel, alongside the conventional i.i.d. initialization baseline.

The results are averaged over all 83 UCI benchmark datasets used in our experimental evaluation. Table 15 summarizes the average training time across datasets.

As observed, all copula-based initializations introduce only a marginal computational overhead relative to the i.i.d. baseline. The additional cost stems primarily from the estimation of the empirical copula parameters and subsequent dependency-aligned sampling. However, since these operations scale linearly with the number of features and hidden nodes, the overall training complexity remains effectively unchanged in practice.

Importantly, the empirical overhead remains below 0.003 seconds on average, confirming that CAWI preserves the computational efficiency characteristic of randomized neural networks while improving stability and performance.

Table 12: BreaKHis dataset (Seed 123): Test accuracy (%). Best per row is **bold** and highlighted; the last column reports the improvement over the i.i.d. baseline.

Dataset	iid	Gaussian	t	Clayton	Frank	Gumbel	Improvement
AD vs. DC	70.3251	**73.1089**	72.5614	69.9123	70.2947	71.3458	**+2.7838**
AD vs. LC	61.8742	63.2184	**64.8936**	62.7451	63.4826	63.1572	**+3.0194**
AD vs. MC	64.1827	65.3636	**66.9091**	64.7273	65.4545	65.0909	**+2.7264**
AD vs. PC	60.8934	61.7238	62.5481	61.1904	62.0917	**63.1549**	**+2.2615**
FA vs. DC	63.9128	**65.7303**	65.1685	65.0562	64.5843	64.4157	**+1.8175**
FA vs. LC	66.2419	66.5783	**67.6194**	66.3891	66.2108	66.5964	**+1.3775**
FA vs. MC	59.4876	59.8512	60.5918	60.6139	**61.3284**	60.8473	**+1.8408**
FA vs. PC	67.6267	67.9733	**69.3200**	66.9067	67.4933	67.7200	**+1.6933**
PT vs. DC	67.8495	70.5614	71.4583	**72.1047**	70.2136	70.5289	**+4.2552**
PT vs. LC	56.7293	**62.5647**	60.5412	59.3176	59.3529	60.4980	**+5.8354**
PT vs. MC	61.5394	62.5893	62.2461	61.8951	**63.4716**	61.1825	**+1.9322**
PT vs. PC	57.7142	58.3725	59.4314	58.2647	**59.9608**	58.3490	**+2.2466**
TA vs. DC	65.0418	66.2349	66.8293	**67.4264**	66.6172	66.5405	**+2.3846**
TA vs. LC	62.5194	63.1508	**63.9271**	63.3915	62.8104	62.8173	**+1.4077**
TA vs. MC	57.5628	59.7834	60.0941	58.7921	**61.1248**	58.8104	**+3.5620**
TA vs. PC	56.7049	56.9863	58.3207	57.6851	59.4632	**61.6894**	**+4.9845**

Table 13: Schizophrenia ROI dataset (Seed 7): Test accuracy (%). Best per row is **bold** and highlighted; the last column reports the improvement over the i.i.d. baseline.

Dataset	iid	Gaussian	t	Clayton	Frank	Gumbel	Improvement
SZ–ROI GM+WM	60.7126	61.9540	**63.4253**	62.8506	62.6897	63.0345	**+2.7127**
SZ–ROI GM	61.9540	63.5632	**63.9080**	62.1839	62.0690	61.4023	**+1.9540**
SZ–ROI WM	61.4483	60.5977	**62.5287**	61.8161	61.2931	59.9425	**+1.0804**

Table 14: Schizophrenia ROI dataset (Seed 123): Test accuracy (%). Best per row is **bold** and highlighted; the last column reports the improvement over the i.i.d. baseline.

Dataset	iid	Gaussian	t	Clayton	Frank	Gumbel	Improvement
SZ–ROI GM+WM	59.9080	61.5402	62.8736	**63.3563**	62.1839	62.7586	**+3.4483**
SZ–ROI GM	62.7586	63.4483	62.6437	62.7126	**64.0230**	61.3793	**+1.2644**
SZ–ROI WM	60.8046	**62.6437**	61.2931	61.8621	60.7816	59.4253	**+1.8391**

Table 15: Average training time (in seconds) across 83 UCI datasets.

Method	i.i.d.	Gaussian	t	Clayton	Frank	Gumbel
Avg. Time (s)	0.00691	0.00824	0.00924	0.00727	0.00733	0.00742