Preference Neural Network

Ayman Elgharabawy, Mukesh Prasad, Senior Member, IEEE, and Chin-Teng Lin, Fellow, IEEE

Abstract—This paper proposes a novel label ranker network to learn the relationship between labels to solve ranking and classification problems. The Preference Neural Network (PNN) uses spearman correlation gradient ascent and two new activation functions, positive smooth staircase (PSS), and smooth staircase (SS) that accelerate the ranking by creating almost deterministic preference values. PNN is proposed in two forms, fully connected three layers and Preference Net (PN), where the latter is the deep ranking form of PNN to learning feature selection using ranking to solve images classification problem. PN uses a new type of ranker kernel to generate a feature map. PNN outperforms five previously proposed methods for label ranking, obtaining state-of-the-art results on label ranking, and PN achieves promising results on CFAR-100 with high computational efficiency.

Index Terms—Preference learning, deep label ranking, neural network.

I. INTRODUCTION

Preference learning (PL) is an extended paradigm in machine learning that induces predictive preference models from experimental data [1], [2], [3]. PL has applications in various research areas such as knowledge discovery and recommender systems [4]. Objects, instances, and label ranking are the three main categories of PL domain. Of those, label ranking (LR) is a challenging problem that has gained importance in information retrieval by search engines [5], [6]. Unlike the common problems of regression and classification [7], [8], [9], [10], [11], [12], [13], label ranking involves predicting the relationship between multiple label orders. For a given instance \( x \) from the instance space \( X \), there is a label \( L \) associated with \( x \), \( \mathcal{L} \subset \pi \), where \( \pi = \{\mathcal{L}_1, \ldots, \mathcal{L}_n\} \), and \( n \) is the number of labels. LR is an extension of multi-class and multi-label classification, where each instance \( x \) is assigned an ordering of all the class labels in the set \( \mathcal{L} \). This ordering gives the ranking of the labels for the given \( x \) object. This ordering can be represented by a permutation set \( \pi = \{1, 2, \ldots, n\} \). The label order has the following three features. irreflexive where \( \lambda_a \neq \lambda_a \), transitive where \( (\lambda_a \succ \lambda_b) \land (\lambda_b \succ \lambda_c) \Rightarrow \lambda_a \succ \lambda_c \) and asymmetric \( \lambda_a \succ \lambda_b \Rightarrow \lambda_b \nRightarrow \lambda_a \). Label preference takes one of two forms, strict and non-strict order. The strict label order \( (\lambda_a \succ \lambda_b \succ \lambda_c \succ \lambda_d) \) can be represented as \( \pi = (1, 2, 3, 4) \) and for non-restricted total order \( \pi = (\lambda_a \succ \lambda_b \simeq \lambda_c \succ \lambda_d) \) can be represented as \( \pi = (1, 2, 3) \), where \( a, b, c, \text{and}, d \) are the label indexes and \( \lambda_a, \lambda_b, \lambda_c \) and \( \lambda_d \) are the ranking values of these labels.

For the non-continuous permutation space, The order is represented by the relations mentioned earlier and the \( \perp \) incomparability binary relation. For example the partial order \( \lambda_a \succ \lambda_b \succ \lambda_d \) can be represented as \( \pi = (1, 2, 0, 3) \) where 0 represents an incomparable relation since \( \lambda_c \) is not comparable to \( (\lambda_a, \lambda_b, \lambda_d) \).

Various label ranking methods have been introduced in recent years [14], such as decomposition-based methods, statistical methods, similarity, and ensemble-based methods. Decomposition methods include pairwise comparison [15], [16], log-linear models and constraint classification [17]. The pairwise approach introduced by Hüllermeier [18] divides the label ranking problem into several binary classification problems to predict the pairs of labels \( \lambda_i \succ \lambda_j \) or \( \lambda_j \sim \lambda_i \) for an input \( x \). Statistical methods includes decision trees [19], instance-based methods (Plackett-Luce) [20] and Gaussian mixture model based approaches. For example, Mihaljoc uses Gaussian mixture models to learn soft pairwise label preferences [21].

The artificial neural network (ANN) for ranking was first introduced as (RankNet) by Burge to solve the problem of object ranking for sorting web documents by a search engine [22]. Ranknet uses gradient descent and probabilistic ranking cost function for each object pair. The multilayer perceptron for label ranking (MLP-LR) [23] employs a network architecture using a sigmoid activation function to calculate the error between the actual and expected values of the output labels. However, It uses a local approach to minimize the individual error per output neuron by subtracting the actual-predicted value and using Kendall error as a global approach. Neither direction uses a ranking objective function in backpropagation (BP) or learning steps.

The deep neural network (DNN) is introduced for object ranking to solve document retrieval problems. RankNet [22], RankBoost [24], and Lambda MART [25], and deep pairwise label ranking models [26], are convolution neural Network (CNN) approaches for the vector representation of the query and document-based. CNN is used for image retrieval [27] and label classification for remote sensing and medical diagnosing [28].
A multi-valued activation function has been proposed by Moraga and Heider [36] to propose a Generalized Multiple-valued Neuron with a differentiable soft staircase activation function, which is represented by a sum of a set of sigmoidal functions. In addition, Aizenberg proposed a generalized multiple-valued neuron using a convex shape to support complex numbers neural network and multi-values numbers [37]. Visual saliency detection using the Markov chain model is one approach that simulates the human visual system by highlighting the most important area in an image and calculating superpixels as absorbing nodes [38], [39], [40]. However, this approach needs a saliency optimization on the results and has calculation cost [41], [42].

Particle Swarm Optimization in movement detection is based on the concept of variation and inter-frame difference for feature selection. The swarm algorithms are mainly used in human motion detection in sports, and it is used based on probabilistic optimization algorithm [43], [44], [45], [46] and CNN [47].

Some of the methods mentioned above and their variants have some issues that can be broadly categorized into three types:

1) The ANN Predictive probability can be enhanced by limiting the output ranking values in the SS functions to a discrete value instead of a range of values of the rectified linear unit (Relu), Sigmoid, or Softmax activation functions. The predictive is enhanced by using the SS function slope as a step function to create discrete values, accelerating the learning by reducing the output values to accelerate the ranking convergence.

2) The drawback of ranking based on the classification technique ignores the relation between multiple labels: When the ranking model is constructed using binary classification models, these methods cannot consider the relationship between labels because the activation functions do not provide deterministic multiple values. Such ranking based on minimizing pairwise classification errors differs from maximizing the label ranking’s performance considering all labels. This is because pairs have multiple models that may reduce ranking unification by increasing ranking pairs conflicts where there is no ground truth, which has no generalized model to rank all the labels simultaneously. For example, \( D = (1, 1, 1) \) for \( \pi = (\lambda_a > \lambda_b > \lambda_c) \) and \( D = (1, 1, 1) \) for \( \pi = (\lambda_a > \lambda_c > \lambda_b) \) the ranking is unique; however, pairwise classification creates no ground truth ranking for the pair \( \lambda_a > \lambda_c \) and \( \lambda_c > \lambda_b \) which adds more complexity to the learning process.

3) Ignoring the relation between features. The convolution kernel has a fixed size that detects one feature per kernel. Thus, it ignores the relationship between different parts of the image. For example, CNN detects the face by combining features (the mouth, two eyes, the face oval, and a nose) with a high probability of classifying the subject without learning the relationship between these features. For example, the proposed PN kernel start attention to the important features that have a high number of pixel ranking variation.

The main contribution of the proposed neural network is

- Solving the label ranking as a machine learning problem.
- Solving the deep learning classification problem by employing computational ranking in feature selection and learning.

Where PNN has several advantages over existing label ranking methods and CNN classification approaches.

1) PNN uses the smooth staircase SS as an activation function that enhances the predictive probability over the sigmoid and Softmax due to the step shape that enhances the predictive probability from a range from \(-1\) to \(1\) in the sigmoid to almost discrete multi-values.

2) PNN uses gradient ascent to maximize the spearman ranking correlation coefficient. In contrast, other classification-based methods such as MLP-LR and RankNet use the absolute difference of root mean square error (RMS) by calculating the differences between actual and predicted ranking and other RMS optimization, which may not give the best ranking results.

3) PNN is implemented directly as a label ranker. It uses staircase activation functions to rank all the labels together in one model. The SS or PSS functions provide multiple output values during the conversions; however, MLP-LR and RankNet use sigmoid and Relu activation functions. These activation functions have a binary output. Thus, it ranks all the labels together in one model instead of pairwise ranking by classification.

4) PN uses a novel approach for learning the feature selection by ranking the pixels and using different sizes of weighted kernels to scan the image and generate the features map.

The next section explains the Ranker network experiment, problem formulation, and the PNN components (Activation functions, Objective function, and network structure) that solve the Ranker problems and comparison between Ranker network and PNN.

II. PNN COMPONENTS

A. Initial Ranker

The proposed PNN is based on an initial experiment to implement a computationally efficient label ranker network based on the Kendall \( r \) error function and sigmoid activation function using simple structure as illustrated in Section IV Fig. 6.

The ranker network is a fully connected, three-layer net. The input represents one instance of data with three inputs, and there are six neurons in the hidden layer and three output neurons representing the labels’ index. Each neuron represents the ranking value. A small toy data set is used in this experiment. The ranker uses RMS gradient descent as an error function to measure the difference between the predicted and actual ranking values. The ranker has Kendall \( r \) as a stopping criterion. The same ANN structure, number of neurons and learning rate using SS activation function, and spearman error function and gradient ascent of \( \rho \) will be discussed in Section IV. The ranking convergence reaches \( r \approx 1 \) after 160 epochs using the Sigmoid function [48]. The sigmoid and ReLU shapes have a slightly high rate of change of \( y \), and it produces a larger output range.
of data. Therefore, we consider ranking performance as one of the disadvantages of sigmoid function in the ranker network.

The ranker network has two main problems.

1) The ranker uses two different error functions, RMS for learning and Kendall \( \tau \) for stopping criteria. Kendall \( \tau \) is not used for learning because it is not continuous or differentiable. Both functions are not consistent as stopping criteria measure the relative ranking, and RMS does not, which may lead to incorrect stopping criteria. Enhancing the RMS may not also increase the error performance, as illustrated in Fig. 3 in a comparison between the ranker network, evaluation using \( \rho \) and RMS.

2) The convergence performance takes many iterations to reach the ranking \( \tau \approx 1 \) based on the shape of sigmoid or ReLu functions and learning rate as shown in the experiment video link [48] due to the slope shape between -1 or 0 and 1. The prediction probability almost equals the values from \(-1\) or \(0\) to \(1\).

B. Problem Formulation

For multi-class and multi-label problems, learning the data’s preference relation predicts the class classification and label ranking, i.e. data instance \( D \in \{x_1, x_2, \ldots, x_n\} \), the output labels are predicted as ranked set labels that have preference relations \( L = \{\lambda_{y_1}, \ldots, \lambda_{y_n}\} \). PNN creates a model that learns from an input set of ranked data to predict a set of new ranked data. The next section presents the initial experiment to rank labels using the usual network structure.

C. Activation Functions

The usual ANN activation functions have a binary output or range of values based on a threshold. However, these functions do not produce multiple deterministic values on the \( y \)-axis. This paper proposes new functions to slow the differential rate around ranking values on the \( y \)-axis to solve ranking instability. The proposed functions are designed to be non-linear, monotonic, continuous, and differentiable using a polynomial of the \( \text{tanh} \) function. The step width maintains the stability of the ranking during the forward and backward processes. Moraga [36] introduced a similar multi-valued function. However, the proposed exponential derivative was not applied to an ANN implementation. Moraga exponential function is geometrically similar to the step function [49]. However, the newly proposed functions consist of \( \text{tanh} \) polynomial instead of exponential due to the difficulty in implementation. The new functions detect consecutive integer values, and the transition from low to high rank (or vice versa) is fast and does not interfere with threshold detection.

1) Positive Smooth Staircase (PSS): As a non-linear and monotonic activation function, a positive smooth staircase (PSS) is represented as a bounded smooth staircase function starting from \( x = 0 \) to \( \infty \). Thus, it is not geometrical symmetrical around the \( y \)-axis as shown in Fig. 1. PSS is a polynomial of multiple \( \text{tanh} \) functions and is therefore differentiable and continuous. The function squashes the output neurons values during the \( FF \) into finite multiple integer values. These values represent the preference values from \( \{0 \text{ to } n\} \) where \( 0 \) represents the incomparable relation \( \perp \) and values from \( 1 \) to \( n \) represent the label ranking. The activation function is given in (1). PSS is scaled by increasing the step width \( w \)

\[
y = -\frac{1}{2s} \left( \sum_{i=0}^{n-1} \text{tanh} \left(c(wi - x)\right) - n \right)
\]

where \( n \) is the number of stair steps equal to the number of labels to rank, \( w \) is the step width, and \( c \) is the stair curvature \( c = 100 \) and \( 5 \) for the sharp and smooth step, respectively, and \( s \) is the scaling factor for reducing the height of each step to range to rank value with decimal place for the regression problems. \( s = 10 \) and \( s = 100 \) for 1 and 2 decimal places, respectively, \( s \) is calculated as in (2).

\[
n = Y_{\text{max}}s
\]

and \( w \) is the step width as shown in (3).

\[
2b = w(n - 1)
\]
The activation function is given in (4).

\[ y = -\frac{1}{2s} \left( \sum_{i=0}^{n-1} \tanh \left( c \left( b - x - w_i \right) \right) - n \right) \]  

(4)

where \( c \) is step curvature, \( n \) = number of ranked labels, \( b \) is the boundary value on the x-axis, and \( (SS) \) lies between \( -b \) and \( b \). where \( Y_{\text{max}} \) is the max. value to rank, i.e. \( Y_{\text{max}} = 3 \) and values have one decimal place. \( n = 30 \) The \( (SS) \) function has the shape of smooth stair steps, where each step represents an integer number of label ranking on the y-axis from \( 0 \) to \( \infty \) as shown in Fig. 1. The SS step is not flat, but it has a differential slope. The function boundary value on the x-axis is from \( -b \) to \( b \). Therefore, input values must be scaled from \( -b \) to \( b \). The step width is 1 when \( n \approx 2b \). The convergence rate is based on the step width. However, it may take less time to converge based on network hyper parameters. Fig. 2(a) and (b). The SS is scaled by increasing the boundary value \( b \).

D. Ranking Loss Function

Two main error functions have been used for label ranking; Kendall \( \tau \) [50] and spearman \( \rho \) [51]. However, the Kendall \( \tau \) function lacks continuity and differentiability. Therefore, the spearman \( \rho \) correlation coefficient is used to measure the ranking between output labels. spearman \( \rho \) error derivative is used as a gradient ascent process for BP, and correlation is used as a ranking evaluation function for convergence stopping criteria. \( \tau_{\text{Avg}} \) is the average \( \tau \) per label divided by the number of instances \( m \), as shown in line 8 of Algorithm 1. spearman \( \rho \) measures the relative ranking correlation between actual and expected values instead of using the absolute difference of root means square error \( (RMS) \) because gradient descent of \( RMS \) may not reduce the ranking error. For example, \( \pi_1 = (1, 2, 1, 2, 2) \) and \( \pi_2 = (1, 2, 2, 1) \), have a low \( RMS = 0.081 \) but a low ranking correlation \( \rho = 0.5 \) and \( \tau = 0.3 \).

Fig. 3 shows the comparison between the initial ranker network and PNN; the ranker network uses Kendall \( \tau \) which has lower performance as a stopping criterion compared to PNN spearman because the stopping criteria are based on the \( RMS \) per iteration; however, PNN uses spearman for both ranking step and stopping criteria.

The spearman error function is represented by (5)

\[ \rho = 1 - \frac{6 \sum_{i=1}^{m} (y_i - y_{\text{t}_i})^2}{m(m^2 - 1)} \]  

(5)

where \( y_i, y_{\text{t}_i}, i \) and \( m \) represent rank output value, expected rank value, label index and number of instances, respectively.

E. PNN Structure

1) One Middle Layer: The ANN has multiple hidden layers. However, we propose PNN with a single middle layer instead of multi-hidden layers because ranking performance is not enhanced by increasing the number of hidden layers due to fixed multi-valued neuron output, as shown in Fig. 4; Seven benchmark data sets [52] was experimented using \( SS \) function

Fig. 2. SS activation function where \( n = 6, 30 \) and 20 and boundary \( b = 1, 30 \) and 1 and scale factor for the decimal place is \( s = 1, 1 \) and 10 for ranking/classification, extreme label ranking/classification and regression in (a), (b) and (c) respectively.

Fig. 3. Ranker network and PNN evaluation in terms of \( RMS \) and spearman correlation error functions.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Fig. 4. Multiple layer label ranking comparison of benchmark data sets [52] results using the PNN and SS functions after 100 epochs and learning rate = 0.007.

Fig. 5. The structure of preference neuron where $\phi_n = 4$.

using one, two, and three hidden layers with the following hyper parameters; learning rate (l.r.) = 0.05, and each layer has neuron $i = 100$ and $b = 10$). We found that by increasing the number of hidden layers, the ranking performance decreases, and more iterations are required to reach $\rho \approx 1$. The low performance because of the shape of SS produces multiple deterministic values, which decrease the arbitrarily complex decision regions and degrees of freedom per extra hidden layer.

2) Preference Neuron: Preference Neuron are a multi-valued neurons uses a PSS or SS as an activation function. Each function has a single output; however, PN output is graphically drawn by $n$ number of arrow links that represent the multi-deterministic values. The PN in the middle layer connects to only $n$ output neurons $st_p = n + 1$; where $st_p$ is the number of SS steps. The PN in the output layer represents the preference value. The middle and output PNs produce a preference value from 0 to $\infty$ as illustrated in Fig. 5.

The PNN is fully connected to multiple-valued neurons and a single-hidden layer ANN. The input layer represents the number of features per data instance. The hidden neurons are equal to or greater than the number of output neurons, $H_n \geq L_n$, to reach error convergence after a finite number of iterations. The output layer represents the label indexes as neurons, where the labels are displayed in a fixed order, as shown in Fig. 6.

The ANN is scaled up by increasing the hidden layers and neurons; however, increasing the hidden layers in PNN does not enhance the ranking correlation because it does not arbitrarily increase complex decision regions and degrees of freedom to solve more complex ranking problems. This limitation is due to the multi-semi discrete-valued activation function, limiting the output data variation. Therefore, instead of increasing the hidden layer, PNN is scaling up by increasing the number of neurons in the middle layer and scaling input data boundary value and increasing the PSS step width and SS boundaries which are equal to the input data scaling value, which leads to increased data separability.

PNN reaches ranking $\rho \approx 1$ after 24 epochs compared to the initial ranker network that reaches the same result in 200 iterations. The video link demonstrates the ranking convergence as shown in Fig. 7 and video [48]. A summary of the three networks is presented in Table I.

The output labels represent the ranking values. The differential PSS and SS functions to accelerate the convergence after a few iterations due to the staircase shape, which achieves stability in learning. PNN simplifies the calculation of FF and BP, and updates weights into two steps due to single middle layer

| Type | Ranker ANN | PNN |
|------|------------|-----|
| Activation Fun. | ReLU/Sigmoid | PSS, SS |
| Gradient | Descent | Ascent |
| Objective Fun. | RMS | $\rho$ |
| Stopping Criteria. | $t$ | $\rho$ |
Fig. 7. The structure used in both ranker ANN and PNN where \( \phi_n = 3 \), \( f_{in} = 3 \), \( \lambda_{out} = 3 \) per \( \pi_1 = \{1, 2, 3\} \), and comparison of the convergence for both NN’s. The demo video of convergence of two NN in the link [48].

architecture. Therefore, the batch weight updating technique is not used in PNN, and pattern update is used in one step. The network bias is low due to the limited preference neuron output of data variance; thus it is not calculated. Each neuron uses the SS or PS activation function in FF step, and calculates the preference number from 1 to \( n \), where \( n \) is the number of label classes. During BP, The processes of FF and BP are executed in two steps until \( \rho_{Avg} \approx 1 \) or the number of iterations reaches \( 10^6 \) as mentioned in the algorithm section.

The SS step width decreases by increasing the number of labels; thus, we increase function boundary \( b \) to increase the step width to \( \approx 1 \) to make the ranking convergence; In addition, a few complex data sets may need more data separability to enhance the ranking. Therefore, we use the \( b \) value as a hyperparameter to keep the stair width \( > 1 \) and normalize input data from \(-b\) to \( b\).

Table I shows a brief comparison between Ranker ANN and PNN.

The following section describes the data preprocessing steps, feature selections, and components of PN.

III. PN COMPONENTS

A. Image Preprocessing

1) Greyscale Conversion: Data scaling as red, green, and blue (RGB) colors is not considered for ranking because PN measures the preference values between pixels. Thus, The image is converted from RGB color to Greyscale.

2) Pixels’ Sorting: Ranking the image from \( \pi = \{\lambda_1, \ldots, \lambda_m\} \) to \( \pi = \{\lambda_1, \ldots, \lambda_k\} \) where the maximum greyscale value \( \lambda_m = 255 \) and \( \lambda_k \) is the maximum ranked pixel value as illustrated in Fig. 8(a).

3) Pixels Averaging: Ranking image pixels has an almost low ranking correlation due to noise, scaling, light, and object movement; therefore, window averaging is proposed by calculating the mean of pixel values of the small flattened window size of \( 2 \times 2 \) of 4 pixels as shown in Fig. 9. The overall image \( \rho \) of pixels increased from 0.2 to 0.79 in (a and b), from 0.137 to 0.75 for noisy images in (s and d), and scaled images from \(-0.18\) to 0.71 in (e and f).

The two approaches, Pixel ranking and Averaging has been tested in remote sensing and faces images to detect the similarity, and it shows high ranking correlations using different window size as shown in Fig 10. It detects the high correlation by starting from the large window size = image size. It reduces the size and scans until it reaches the highest correlation.

B. Feature Selection by Attention

Feature selection for the kernel proceeded by selecting the features with a high group of pixel ranking variations indicating the importance of the scanned kernel area. This kind of hard
attention makes the selection based on the threshold of pixel ranking values. to reduce the dimension of the input image.

C. Feature Extraction

This paper proposes a new approach for image feature selection based on the preference values between pixels instead of the convolution of pixels array as implemented in CNN. The PN’s features are based on ranking computational space. Therefore, the kernel window size is considered a factor for feature selection.

1) Pixels Resorting: The flattened window’s values are sorted for each kernel window in the image. The Fig. 8(b) shows the window size $3 \times 3$ range from $\lambda_{k_1} = 23$ to $\lambda_{k_2} = 9$. Pixel sorting reduces the data margin, Thus, it reduces the computational complexity.

2) Weighted Ranker Kernel: The kernel weights are randomly initialized from $-0.05$ to $0.05$. The kernel learns the features by BP of its weights to select the best feature. the partial change in the kernel is calculated by differentiating the spearman correlation as in (6)

$$dKw = 2 \cdot Imgw - d\rho \cdot \frac{n^3 - n}{6}$$ (6)

Different kernel sizes could be used for big images’ size. We use three different kernels to capture the relations between different features.

3) Max Pooling: Max, pooling is used to reduce the features map’s size and select the highest correlation values to feed to the PNN.

D. PN Structure

PN is the deep learning structure of PNN for image classification. It consists of five layers, a ranking features map and a max. pooling and three PNN layers. PN has one or multiple different sizes of PNNs connected by one output layer. Each PNN has SS or PSS where $\varphi_{n=2}$ for binary ranking to map the classification. The number of output neurons is the number of classes. The structure is shown in Fig 11. PN have one or more ranker kernels with different sizes, Each kernel has one corresponding PNN. PN uses the weighted kernel ranking to scan the image and extract the features map of spearman correlation values of the kernel with the scanned ranked image window as $\rho(\pi_k, \pi_w)$ where $\pi_k$ is the kernel preference values and $\pi_w$ is the scanned window image preference values. Each kernel scans the image by one step and creates a spearman features list. Max. Pooling is used to minimize the feature map used as input to PNN.

One 5X5 kernel is used for fashion Mnist data set [53]. Three kernels with sizes (3, 10, and 20) are used for CFAR-100 [54].

E. Choosing the Kernel Size

Kernel size is chosen based on the hard attention of the highest group of pixels that has high ranking variation. The process scans the image sequentially starting from a small size to find the size with the highest pixels ranking variation. For example for the Mnist dataset where the image has a size of $28 \times 28$, The meaningful features are extracted using kernel sizes $10 \times 10$, $15 \times 15$, $20 \times 20$ and $25 \times 25$.

IV. ALGORITHMS

A. Baseline Algorithm

Algorithm 1 represents the three functions of the network learning process; feed-forward (FF), BP, and updating weights (UW). Algorithm 2 represents the learning flow of PN. Algorithm 3 represents the simplified BP function in two steps.

B. Ranking Visualization

PNN ranking convergence is visualized using the SS function by displaying the normalized input data points with corresponding actual ranked five labels represented in 5 different colours, The plotting of input value and SS values against the actual colour values at iterations 0 and 3900 is shown in Fig. 12, which illustrates the distribution of SS output values against the actual colour values at iterations 0 and 3900 and $\tau$ is enhanced from 0.39 to 0.85.

C. Complexity Analysis

1) Time Complexity:

- **FF** time complexity corresponds to FF of middle and output layers, and $m$ and $n$ are the number of nodes in the middle and output layers. $W_m$ and $W_o$ are weighted matrix and $SS_i$ is the activation function of number of instances $t$. The time complexity in (7)

$$O(m \cdot o \cdot t)$$ (7)

- **BB** starts with calculating the error of output layer $E_{ot} = \rho_{o'}Delta_o = E_{ot} \cdot SS'$ and $Delta_m = E_{mt} \cdot SS'$

Algorithm 1: PN Learning flow.

| Data: $D \in \{x_1, x_2, \ldots, x_d\}$ |
|-----------------------------------------|
| Result: $\pi \in \{\lambda_{x_1}, \lambda_{x_2}\}$ |
| 1) Randomly initialize weights $\omega_{i,j} \in (-0.05, 0.05)$ |
| 2) repeat |
| 3) forall $(x_i, \pi_i) \in D$ do |
| 4) $a_i|_{t-1} = \sum_{1}^{m} \varphi(a_i \cdot \omega_{i})_n // FF$ |
| 5) PNN BP() |
| 6) $\omega_{new} = \omega_{old} - \eta \cdot \delta_{i} // UW$ |
| 7) until $\rho_{avg} = 1$ or $\# \text{iterations} \geq 10^6$ |

Fig. 10. Detecting the similarity in remote sensing and face recognition by ranking the image pixels after averaging the pixels using a 2x2 window.
Fig. 11. The PN structure has three kernels and three PNNs where \( \phi_n = 2 \), \( f_{1 in} = 16 \), \( f_{2 in} = 81 \), and \( f_{3 in} = 169 \) and \( \lambda_{out} = 15 \). Fig. 13 shows the convergence after 500 iterations using four activation functions (SS, PSS, sigmoid, and tanh) respectively. We noticed that PSS and SS have a stable rate of ranking convergence compared to sigmoid, tanh, and ReLU. This stability is due to the stairstep width, which leads each point to reach the correct ranking during FF and BP in fewer epochs.

A. Activation Functions Evaluation

PNN is tested on iris and stock data sets using four activation functions. SS, PSS, ReLU, sigmoid, and tanh. PNN has one middle layer and the number of hidden neurons (h.n.) is 50, while l.r. = 0.05. Fig. 13 shows the convergence after 500 iterations using four activation functions (SS, PSS, sigmoid, ReLU and tanh) respectively. We noticed that PSS and SS have a stable rate of ranking convergence compared to sigmoid, tanh, and ReLU. This stability is due to the stairstep width, which leads each point to reach the correct ranking during FF and BP in fewer epochs.

2) Input Neurons: The number of PN input neurons is represented by (10)

\[
\#Input = (\text{Img}_w - K_w + 1) \cdot (\text{Img}_h - K_h + 1) \tag{10}
\]

where \( w \) and \( h \) are width and height of kernel and image.

V. NETWORK EVALUATION

This section evaluates the PNN against different activation functions and architectures. All weights are initialized = 0 to compare activation functions and A and B have the same initialized random weights to evaluate the structure.

This time complexity is then multiplied by the number of epochs \( p \)

\[
O(p \cdot m \cdot o \cdot t) \tag{9}
\]
Algorithm 2: PN Learning flow.

1. Converting image to greyscale
2. Flattening image
3. Pixel sorting
4. 2D Image
5. Pixel averaging by a 2x2 window
6. Flattening image
7. Select one/more kernel sizes.
8. Random init. Kernel $K_{x,y} \in \{-0.05, 0.05\}$
9. Random init. $PNN \omega_{i,j} \in \{-0.05, 0.05\}$
10. repeat
11. 2D Image
12. Scanned window pixel ranking $Img_w$
13. Compute $\rho(Img_w, Kw)$ feature map
14. Max. Pooling
15. Flattening image
16. PNN FF()
17. PNN BP()
18. PNN UW()
19. Max. Pooling BP()
20. Ranker kernel BP and UW()
21. until $\rho_{Avg} = 1$ or #iterations $\geq 10^6$

Algorithm 3: PNN BP.

1. for each $p_n_i$ in Output layer do
2. $Err_1 = \rho = -6 \cdot \frac{(2y_i - y_i)}{m(n^2 - 1)}$ //spearman error
3. $\delta_1 = Err \cdot \phi t$
4. for each $p_n_i$ in middle layer do
5. $Err_1 = \sum_{k=0}^{m} \omega_k \cdot \delta_k$
6. $\delta_1 = Err \cdot \phi t$

2) Missing Labels Evaluation: Activation functions are evaluated by removing a random number of labels per instance. PNN marked the missing label as $-1$; PNN neglects error calculation during $BP$, $\delta = 0$. Thus, the missing label weights remain constants per learning iteration. The missing label approach is applied to the data set by 20% and 60% of the training data. The ranking performance decreases when the number of missing labels increases. However, SS and PSS have more stable convergence than other functions. This evaluation is performed on the iris data set, as shown in Fig. 13.

3) Statistical Test: The PNN results were evaluated using receiver operating characteristic (ROC) curves. The true positive and negative for each rank are evaluated per label of wine dataset as shown in Fig. 14. The confusion matrix on wine and glass DS are shown in Fig. 15 where $\tau = 0.947$, 0.84, Accuracy $= 0.935$ and 0.8 in (a) and (b) respectively.

4) Dropout Regularization: Dropout is applied as a regularization approach to enhance the PNN ranking stability by reducing over-fitting. We drop out the weights that have a probability of less than 0.5. These dropped weights are removed from FF, BP, and UW steps. The comparison between dropout and non-dropout of $PNN$ are shown in Fig. 16. The gap between the training model and ten-fold cross-validation curves has been reduced using dropout regularization using hyperparameters (l.r. $= 0.05$, h.n. $= 100)$ on the iris data set. The dropout technique is used with all the data ranking results in the next section.

The following section is the evaluation of ranking experiments using label benchmark data sets.

VI. EXPERIMENTS

This section describes the classification and label ranking benchmark data sets, the results using $PN$ and $PNN$, and a comparison with existing classification and ranking methods.

A. Data Sets

1) Image Classification Data Sets: $PN$ is evaluated using CFAR-100 [54] and Fashion-MNIST [55] data sets.

2) Label Ranking Data Sets: $PNN$ is experimented with using three different types of benchmark data sets to evaluate the
multi-label ranking performance. The first type of data set focuses on exception preference mining [56], and the ‘algae’ data set is the first type that highlights the indifference preferences problem, where labels have repeated preference value [57]. German elections 2005, 2009, and modified sushi are considered new and restricted preference data sets. The second type is real-world data related to biological science [18]. The third type of data set is semi-synthetic (SS) taken from the KEBI Data Repository at the Philipps University of Marburg [52]. All data sets do not have ranking ground truth, and all labels have a continuous permutation space of relations between labels. Table II summarizes the main characteristics of the data sets.

B. Results

1) Image Classification Results: PN has 3 kernel sizes of 5,10 and 20 and is tested on the CFAR-100 [54] data set and 1 kernel with a size 5 for Fashion-MNIST data set [55]. Table IV shows the results compared to other convolutions networks.

2) Label Ranking Results: PNN is evaluated by restricted and non-restricted label ranking data sets. The results are derived using spearman $\rho$ and converted to Kendall $\tau$ coefficient for comparison with other approaches. For data validation, we used

Fig. 13. PNN activation function comparison using complete labels and 60% missing labels in (a) and (b), respectively.

Fig. 14. ROC of three labels ranking on the wine data set using PNN $h.n = 100$ and 50 epochs.

Fig. 15. The confusion matrix of testing the wine, glass data sets where $\tau = 0.947, 0.84$, Accuracy $= 0.935$ and 0.8 in (a) and (b) respectively.

Fig. 16. Training and validation performance without and with dropout regulation approach in (a) and (b) respectively.
10-fold cross-validation. To avoid the over-fitting problem, we used hyperparameters, i.e. l.r. = (0.0008, 0.0005, 0.005, 0.05, 0.1) hidden neuron = no.inputs + (5, 10, 50, 100, 200, 300, 400, 450) neurons and scaling boundaries from 1 to 250) are chosen within each cross-validation fold by using the best l.r. on each fold and calculating the average $\tau$ of ten folds. Grid searching is used to obtain the best hyperparameter. For type $B$, we use three output groups and l.r. = 0.001 and $w_b = 0.01$.

3) Benchmark Results: Table V summarizes PNN ranking performance of 16 strict label ranking data sets by l.r. and m.n. The results are compared with the four methods for label ranking; supervised clustering [58], supervised decision tree [52], MLP label ranking [23], and label ranking tree forest (LRT) [67]. Each method’s results are generated by ten-fold cross-validation. The comparison selects only the best approach for each method.

During the experiment, it was found that ranking performance increases by increasing the number of central neurons up to a maximum of 20 times the number of features. As shown in Table VI, The real datasets are ranked using PNN with dropout regulation due to complexity and over-fitting. The dropout requires increasing the number of epochs to reach high accuracy. All the results are held using a single hidden layer with various hidden neurons (100 to 450) and SS activation function. The Kendall $\tau$ error converges and reaches close to 1 after 2000 iterations, as shown in Fig. 17.

Table V compares PNN with similar approaches used for label ranking. These approaches are; Decision trees [58], MLP-LR [23] and label ranking trees forest LRT [67]. In this comparison, we choose the method that has the best results for each approach.

4) Preference Mining Results: The ranking performance of the new preference mining data set is represented in Table III. Two hundred fifty hidden neurons are used To enhance the ranking performance of the algae data set’s repeated label values. However, restricted labels ranking data sets of the same type, i.e., (German elections and sushi), did not require a high number of hidden neurons and incurred less computational cost.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Fig. 17. Ranking performance comparison of PNN with other approaches.

TABLE VI

| Type            | DS  | Avg. τ | #m.n. | Lr. | #Iterations | Dropout | Scaling. | Training t. | Testing t. |
|-----------------|-----|--------|-------|-----|-------------|---------|----------|--------------|-------------|
| Real            | cold | 0.4    | 10    | 0.0008 | 2000 | yes | -4.4 | 2.8h | 1.2s |
|                 | dian | 0.466  | 400   | 0.00005 | 2500 | yes | -2.2 | 2.8h | 4s |
|                 | heat | 0.876  | 450   | 0.00005 | 5000 | yes | -2.2 | 6.2h | 1.1s |
|                 | spo  | 0.8    | 300   | 0.00005 | 5000 | yes | -4.4 | 7.4h | 0.98s |
|                 | German2005 | 0.8 | 300  | 0.00005 | 1000 | no | -4.4 | 35.15m | 0.0879s |
|                 | German2009 | 0.67 | 300   | 0.00005 | 500 | no | -4.4 | 7.087m | 0.105s |
| Semi-Synthesized | authorship | 0.931 | 200  | 0.00008 | 200 | no | -4.4 | 5.82m | 0.34s |
|                 | bodyfat | 0.859 | 100  | 0.00005 | 2500 | yes | -2.2 | 16.92m | 0.44s |
|                 | carhousing | 0.34 | 200  | 0.00007 | 1000 | no | -2.2 | 5.03h | 4.127s |
|                 | cpu-small | 0.46 | 200  | 0.005 | 1000 | no | -2.2 | 2.089h | 1.717h |
|                 | elevators | 0.73 | 20  | 0.003 | 100 | no | -2.2 | 27.03m | 3.7s |
|                 | fried | 0.89 | 100  | 0.005 | 100 | no | -2.2 | 1.02h | 8.45s |
|                 | glass | 0.948 | 100  | 0.005 | 100 | no | -3.3 | 14.8s | 0.04s |
|                 | housing | 0.7615 | 25  | 0.005 | 100 | no | -3.3 | 37.21s | 0.1s |
|                 | iris | 0.986 | 100  | 0.005 | 100 | no | -3.3 | 29.39s | 0.066s |
|                 | digits | 0.86 | 100  | 0.005 | 100 | no | -3.3 | 34.6m | 5.69s |
|                 | segment | 0.956 | 200  | 0.007 | 1000 | no | -3.3 | 44.08s | 0.94s |
|                 | stock | 0.868 | 100  | 0.005 | 100 | no | -3.3 | 142.48s | 0.87s |
|                 | vehicle | 0.869 | 100  | 0.005 | 100 | no | -3.3 | 91s | 0.2s |
|                 | vowel | 0.85 | 100  | 0.005 | 100 | no | -3.3 | 88.37s | 0.312s |
|                 | wine | 0.90 | 100  | 0.005 | 100 | no | -3.3 | 19.19s | 0.063s |
|                 | wisconsin | 0.61 | 300  | 0.00005 | 2500 | yes | -4.4 | 13.58m | 0.1332s |

Experiments on the real-world biological data set were conducted using supervised clustering (SC) [58]. Table V presents the comparison between PNN and supervised clustering on biological real world data in terms of $Loss_{LR}$ as given in (11).

$$\tau = 1 - 2 \cdot Loss_{LR}$$

(11)

where $\tau$ is Kendall $\tau$ ranking error and $Loss_{LR}$ is the ranking loss function.

SS function with 16 steps is used to rank Wisconsin data set with 16 labels. By increasing the number of steps in the interval and scaling up the features between $-100$ and $100$, The step width is small. To enhance ranking performance, the data set has many labels. The number of hidden neurons is increased to exceed $\tau = 0.5$.

C. Computational Platform

PNN and PN is implemented from scratch without the Tensorflow API and developed using Numba API to speed the execution on the GPU and use Cuda 10.1 and Tensorflow-GPU 2.3 for GPU execution and executed at the University of Technology Sydney High-Performance Computing cluster based on Linux RedHat 7.7, which has an NVIDIA Quadro GV100 and memory of 32 G.B. For a non-GPU version of PNN is located at GitHub Repository [68].

D. Discussion and Future Work

It can be noticed from Table II that PN is performing better than ResNet [59] and WRN [60]. Different types of architectures of $P\!N\!N$ could be used to enhance the results and reach...
It can be noticed from Table III that PNN outperforms on SS data sets with \( \tau_{AVG} = 0.8 \), whereas other methods such as, supervised clustering, decision tree, MLP-ranker and LRT, have results \( \tau_{AVG} = 0.79, 0.73, 0.62, 0.475 \), respectively. Also, the performance of PNN is almost 50% better than supervised clustering in terms of ranking loss function \( Loss_{LR} \) on real-world biological data set, as shown in Table V. The superiority of PNN is used for classification and ranking problems. The ranking is used in input data as a feature selection criteria is a novel approach for deep learning.

Encoding the labels’ preference relation to numeric values and ranking the output labels simultaneously in one model is an advanced step over pairwise label ranking based on classification. PNN could be used to solve new preference mining problems. One of these problems is incomparability between labels, where Label ranking has incomparable relation \( \perp \), i.e., ranking space \((\lambda_a \succ \lambda_b \perp \lambda_c) \) is encoded to \((1, 2, -1)\) and \((\lambda_a \succ \lambda_b \perp \lambda_d) \) is encoded to \((1, 2, -1, -2)\). PNN could be used to solve new problem of non-strict partial orders ranking, i.e., ranking space \((\lambda_a \succ \lambda_b \succeq \lambda_c) \) is encoded to \((1, 2, 3)\) or \((1, 2, 2)\). Future research may enhance PNN by adding kernel size and SS parameters as part of the deep learning to choose the best kernel size and SS step width, which could enhance the image attention. Modifying PNN architecture by adding bias and solving noisy label ranking problems.

VII. CONCLUSION

This paper proposed a novel method to rank a complete multi-label space in output labels and features extraction in both simple and deep learning. PNN is a new research direction for image recognition based on new kernel and pixel calculations. PNN and PN are native ranker networks for image classification and label ranking problems that uses SS or PSS to rank the multi-label per instance. This neural network’s novelty is a new kernel mechanism, activation, and objective functions. This approach takes less computational time with a single middle layer. It is indexing multi-labels as output neurons with preference values. The neuron output structure can be mapped to integer ranking value; thus, PNN accelerates the ranking learning by assigning the rank value to more than one output layer to reinforce updating the random weights. PNN is implemented using python programming language 3.6 [68], and activation functions are modelled using wolframe Mathematica software [72]. A video demo that shows the ranking learning processing using toy data is available to download [48].

REFERENCES

[1] J. Fmnkranz and E. Hüllermeier, Preference Learning, 1st ed. Berlin, Germany: Springer, 2010.
[2] R. I. Braffman and C. Domshlak, “Preference handling - an introductory tutorial,” AI Mag., vol. 30, pp. 58–86, 2009.
[3] G. Adomavicius and A. Tuzhilin, “Toward the Next Generation of Recommender Systems: A Survey of the State-Of-The-Art and Possible Extensions,” IEEE Trans Knowl. Data Eng., vol. 17, no. 6, pp. 734–749, Jun. 2005.
[4] M. Montaner and B. López, “A Taxonomy of Recommender Agents on the Internet,” Artif. Intell. Rev. vol. 19, pp. 285–330, 2003.
received the B.S. and master’s degrees in computer engineering in 1999, 2000, and 2002, respectively, and the Ph.D. degree in computer science from the University of Technology Sydney, Australia, in 2012. He has been a Program Manager at the Australian Research Council (ARC) and a Senior Researcher at the Australian Institute of Information Technology and Innovation (AIITI). His research interests include neural network design (Neuromorphic Engineering), software engineering, and data science.

[31] Z. Lv, L. Qiao, J. Li, and H. Song, “Deep-learning-enabled security issues in the Internet of Things,” IEEE Internet Things J., vol. 8, no. 12, pp. 9531–9538, Jun. 2021.

[32] Z. Lv, Z. Yu, S. Xie, and A. Alamri, “Deep learning-based smart predictive evaluation for interactive multimedia-enabled smart healthcare,” ACM Trans. Multimedia Comput. Commun. Appl., vol. 18, no. 1s, pp. 1–20, Jan. 2022.

[33] J. Xu, S. Pan, P. Z. H. Sun, S. Heyeong Park, and K. Guo, “Human-factors-in-driving-loop: Driver identification and verification via a deep learning approach using psychological behavioral data,” IEEE Trans. Intell. Transp. Syst., vol. 24, no. 3, pp. 3383–3394, Mar. 2023.

[34] C. Zhan, Z. Dai, M. R. Soltanian, and F. P. J. de Barros, “Data-width analysis for heterogeneous subsurface structure identification with a stochastic deep learning framework,” Water Resour. Res., vol. 58, no. 11, 2022, Art. no. e2022WR033241.

[35] S. Pare et al., “Remote sensing imagery segmentation: A hybrid approach,” Remote Sens., vol. 13, no. 22, 2021, Art. no. 4604. [Online]. Available: https://www.mdpi.com/2072-4292/13/22/4604.

[36] C. Moraga and R. Heider, “New lamps for old!” (generalized multiple-valued neurons),” in Proc. IEEE 29th Int. Symp. Multi.-Valued Log., 1999, pp. 36–41.

[37] I. Aizenberg, N. Aizenberg, and J. P. Vandewalle, Multi-Valued and Universal Binary Neurons: Theory, Learning and Applications. Norwell, MA, USA: Kluwer Academic Publishers, 2000.

[38] W. Zhou, Y. Lv, J. Lei, and L. Yu, “Global and local-contrast guides content-aware fusion for RGB-D saliency prediction,” IEEE Trans. Syst., Man, Cybern. Syst., vol. 51, no. 6, pp. 3641–3649, Jun. 2021.

[39] B. Xie, S. Li, M. Li, C. H. Liu, G. Huang, and G. Wang, “SePSCo: Semantic-guided pixel contrast for domain adaptive segmentation,” IEEE Trans. Syst., Man, Cybern. Syst., vol. 51, no. 6, pp. 3625–3640, Jun. 2021.

[40] A. Elgharabawy, “Preference neural network convergence performance,” 2020. [Online]. Available: https://drive.google.com/drive/folders/1yxuQYo3K3iAch-2LeXZcOcMjJ2QVR7hM?usp=sharing.
Mukesh Prasad (Senior Member, IEEE) received the M.S. degree from the School of Computer Systems and Sciences, Jawaharlal Nehru University, New Delhi, India, in 2009, and the Ph.D. degree from the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, in 2015. He is currently a Senior Lecturer with the School of Computer Science, Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW, Australia. During 2016–2017, he was a Principal Engineer with Taiwan Semiconductor Manufacturing Company, Hsinchu, Taiwan. He started his academic career as a Lecturer with the University of Technology Sydney in 2017. He has authored or coauthored more than 100 articles including several prestigious IEEE Transactions and other Top Q1 journals and conferences in the areas of Artificial Intelligence and Machine Learning. His research expertise lies in the development of new methods in artificial intelligence, machine learning and data analytic approach within the domain of computer vision, healthcare, biomedical, Internet of Things and brain-computer interface and marketing research. He is also an Associate/Area Editor of several top journals in the field of machine learning, computational intelligence, and emergent technologies. He is a Senior Member of ACM.

Chin-Teng Lin (Fellow, IEEE) received the B.S. degree from National Chiao-Tung University, Hsinchu, Taiwan, in 1986, and the Master and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1989 and 1992, respectively. He is currently the Chair Professor with the Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW, Australia, Chair Professor of Electrical and Computer Engineering, NCTU, International Faculty of the University of California at San Diego, San Diego, CA, USA, and Honorary Professorship with the University of Nottingham, Nottingham, U.K. Dr. Lin is the coauthor of Neural Fuzzy Systems (Prentice-Hall), and the author of Neural Fuzzy Control Systems with Structure and Parameter Learning (World Scientific). He has authored or coauthored more than 200 journal papers, Total Citation: 20,155, H-index: 53, i10-index: 373, in neural networks, fuzzy systems, multimedia hardware/software, and cognitive neuro-engineering, including approximately 101 IEEE journal papers. Dr. Lin was elevated to an IEEE Fellow for his contributions to biologically inspired information systems in 2005 and was elevated to International Fuzzy Systems Association (IFSA) Fellow in 2012. Since 2011, he has been elected the Editor-in-chief of IEEE TRANSACTIONS ON FUZZY SYSTEMS. Dr. Lin is the Distinguished Lecturer at the IEEE CAS Society from 2003 to 2005, and CIS Society from 2015 to 2017. From 2006 to 2008, he was the Deputy Editor-in-Chief of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II.