Preference Neural Network

Ayman Elgharabawy∗†, Mukesh Prasad †‡ Senior Member, IEEE, Chin-Teng Lin †‡, Fellow, IEEE
School Of computer Science, University of Technology Sydney (UTS)

Email: *Ayman.Elgharabawy@uts.edu.au, †Mukesh.Prasad@uts.edu.au, ‡Chin-teng.lin@uts.edu.au

∗Corresponding author

Abstract—This paper proposes a novel label ranker network to learn the relationship between labels for classification and ranking 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 simple 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. PNN 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 PNN achieves promising results on CFAR-100 with high computational efficiency.

Index Terms—Preference learning, Deep label ranking, Neural network, Preference mining.

I. INTRODUCTION

Preference learning (PL) is an extended paradigm in machine learning that induces predictive preference models from experimental data [1]–[3]. PL has applications in a variety of 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, label ranking involves predicting the relationship between multiple label orders. For a given instance x from the instance space Ξ, there is a label L associated with x, L ∈ π, where π = {λ1, ..., λn}, 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 L. This ordering gives the ranking of the labels for the given x object. This ordering can be represented by a permutation set π = (1, 2, ..., n). The label order has the following three features. Irreflexive where λa ⊁ λa, transitive where (λa > λb) ∧ (λb > λc) ⇒ λa > λc and asymmetric λa > λb ⇒ λb ⊁ λa. Label preference takes one of two forms, strict and non-strict order. The strict label order (λa > λb > λc > λd) can be represented as π = (1, 2, 3, 4) and for non-restricted total order π = (λa > λb > λc > λd) can be represented as π = (1, 2, 3, 4), where a, b, c, and d are the label indexes and λa, λb, λc, and λd are the ranking values of these labels.

For the non-continuous permutation space, The order is represented by the relations mentioned earlier and the ⋁ incomparability binary relation. For example the partial order λa > λb > λd can be represented as π = (1, 2, 0, 3) where 0 represents an incomparable relation since λc is not comparable to (λa, λb, λd).

Various label ranking methods have been introduced in recent years [7], such as decomposition based methods, statistical methods, similarity, and ensemble-based methods. Decomposition methods include pairwise comparison [8], [9], log-linear models and constraint classification [10]. The pairwise approach introduced by Hüllermeier [11] divides the label ranking problem into several binary classification problems in order to predict the pairs of labels λi > λj or λj < λi for an input x. Statistical methods include decision trees [12], instance-based methods (Plackett-Luce) [13] and Gaussian mixture model [14] based approaches. For example, Mihajlo uses Gaussian mixture models to learn soft pairwise label preferences [14].

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 [15]. RankNet uses gradient descent and probabilistic ranking cost function for each object pair. The multilayer perceptron for label ranking (MLP-LR) [16] 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 [15], RankBoost [17], and Lambda MART [18], and deep pairwise label ranking models [19], are convolution neural Network (CNN) approaches for the vector representation of the query and document-based. CNN is used for image retrieval [20] and label classification [21]. A multi-valued activation function has been proposed by Moraga and Heider [22] 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.

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

1) The ANN prediction probability is almost the output
The proposed PNN label ranking methods and CNN 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 rankings based on minimizing pairwise classification errors are not necessarily equivalent to 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, \( \mathcal{D} = (1,1,1) \) for \( \pi = (\lambda_a > \lambda_b > \lambda_c) \) and \( \mathcal{D} = (1,1,1) \) for \( \pi = (\lambda_a > \lambda_c > \lambda_b) \) the ranking is unique; however, pairwise classification creates no ground truth ranking for the pair \( \lambda_b > \lambda_c \) 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 whole image in large images. For example, CNN detects the face by combining features (the mouth, two eyes, the face oval, and a nose) with a high probability to classify the subject without learning the relationship between these features.

The proposed PNN has several advantages over existing in 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 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) PNN uses a novel approach for learning the feature selection by ranking the pixels and use different size 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 \( \tau \) error function and sigmoid activation function using simple structure as illustrated in section IV Fig. 7.

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 \( \tau \) 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 to \( \tau \approx 1 \) after 160 epochs using the Sigmoid function [23]. The sigmoid and ReLU shape have a slightly high rate of change of \( y \), and it produces 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 both \( \tau \) and RMS and PNN ranking evaluation using \( \rho \) and RMS.

2) The convergence performance takes numbers of 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 [23] due to the slope shape between -1 or 0 and 1. The prediction probability is almost equal the values from -1 or 0 to 1.

B. Problem Formulation

For a multi-class and multi-label problems, learning the data’s preference relation predicts the class classification and label ranking. i.e. data instance \( \mathcal{D} \in \{x_1,x_2,\ldots,x_n\} \).
the output labels are predicted as ranked set labels that have preference relations \( L = (\lambda_{y1}, \ldots, \lambda_{yn}) \). 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 \( \tanh \) function. The step width maintains the stability of the ranking during the forward and backward processes. Moraga [22] 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 [24]. However, the newly proposed functions consist of \( \tanh \) 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 not interfere with threshold detection.

1) **Positive Smooth Staircase (PSS):** As a non-linear and monotonic activation function, positive smooth staircase (PSS) is represented as a bounded smooth staircase function starts from \( x = 0 \) to \( \infty \). Thus, it is not geometrically symmetrical around the \( y \)-axis as shown in Fig. 1. PSS is a polynomial of multiple \( \tanh \) 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 \( 0 \) to \( n \) where \( 0 \) represent the incomparable relation \( \bot \) and values from \( 1 \) to \( n \) represent the label ranking. The activation function is given in Eq. 1. PSS is scaled by increasing the step width \( w \)

![Fig. 1: PSS activation function where \( n = 3 \) and step width \( w = 1 \)](image)

\[
y = -\frac{1}{2} \left( \sum_{i=0}^{n} \tanh(-100(x - wi)) \right) + \frac{n}{2} \tag{1}
\]

Where \( n \) is number of output labels, \( w \) is the step width.

2) **Smooth Staircase (SS):** The proposed (SS) represents a staircase similar to (PSS). However, SS has a variable boundary value used as a hyperparameter in the learning process. The derivative of the activation function is discussed in section III and the performance comparison between SS and PSS is mentioned in section V.

![Fig. 2: SS activation function where \( n = 6 \) and 9 and boundary \( b = 1 \) and 10 in (a) and (b) respectively.

The activation function is given in Eq. 2.

\[
y = -\frac{1}{2} \left( \sum_{i=0}^{n} \tanh(-100(x - wi)) \right) + \frac{n}{2} \tag{2}
\]

where \( c = 100, n = \text{number of ranked labels}, b \) is the boundary value, and (SS) lies between \( -b \) and \( b \). 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 \( x \)-axis is from \(-b\) to \( b \). The step width is 1 when \( n = 2b \). The convergence rate is based on the step width. However, it may take less time to converge based on network hyperparameters. Fig. 2 (a) and (b) shows the activation functions to rank 6 and 9 labels, respectively. 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 \( \tau \) [25] and spearman \( \rho \) [26]. However, the Kendall \( \tau \) function lacks continuity and differentiability. Therefore, The spearman \( \rho \) correlation coefficient is used to measure the ranking between output labels.
\( \rho \) 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. \text{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 \( \pi_2 = (1,2,2,2,1) \), have a low RMS = 0.081 but a low ranking correlation \( \rho = 0.5 \) and \( \tau = 0.3 \).

![Fig. 3: Ranker network and PNN evaluation in terms of RMS and spearman correlation error functions](image)

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

The \text{spearman} error function is represented by Eq.3

\[
\rho = 1 - \frac{6 \sum_{i=1}^{m} (y_i - \lambda y_i)^2}{m(m^2-1)} \tag{3}
\]

where \( y_i, \lambda y_i, i \) 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 [27] was experimented using SS function using one, two, and three hidden layers with the following hyperparameters; 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 iteration are required to reach \( \rho = 1 \). 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.

![Fig. 4: Multiple layer label ranking comparison of benchmark data sets [27] results using the PNN and SS functions after 100 epochs and learning rate = 0.007.](image)

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 arrows links that represent the multi-deterministic values. The PN in the middle layer connects to only \( n \) output neurons \( s t p = n + 1 \); where \( s t p \) is the number of SS steps. The PN in 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, \( H_n \geq \lambda_n \), 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 a more complex ranking problems. This limitation due to the multi-semi discrete-valued activation function, which limits 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 increase 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 demo [23], and a summary of the three networks are presented in Table I.

The output labels represent the ranking values. The differential PSS and SS functions 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 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 PN output variation, so it is not calculated. Each neuron uses the 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 $b$ in order 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 $\geq 1$ and normalize input data from $-b$ to $b$.

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Ranker ANN</th>
<th>PNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activation Func.</td>
<td>ReLU, Sigmoid</td>
<td>PSS, SS</td>
</tr>
<tr>
<td>Gradient</td>
<td>Descent</td>
<td>Ascent</td>
</tr>
<tr>
<td>Objective Func.</td>
<td>RMS</td>
<td>$\rho$</td>
</tr>
<tr>
<td>Stopping Criteria.</td>
<td>$\tau$</td>
<td>$\rho$</td>
</tr>
</tbody>
</table>

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) colours is not considered for ranking because PN measures the preference values between pixels. Thus, the image is converted from RGB colour to Greyscale.

2) Image Pixels' Ranking: Ranking the image from $\pi = \{1, \ldots, \lambda_m\}$ to $\pi = \{1, \ldots, \lambda_k\}$ where the maximum greyscale value $\lambda_m = 255$ and $\lambda_k$ is the maximum ranked pixel value as illustrated in Fig. 8

3) Pixel Averaging: Ranking image pixels has 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 2x2 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 applied on remote sensing and faces images to detect...
The ranking of two flattened images are \( \rho \) without and with image averaging by a window of 2x2. The ranking of two flattened images are \( \rho = 0.216 \) and 0.79 in (a) and (b) respectively. Sample of moving noisy object in (c) and (d) without and with image averaging by a window size of 2x2. The ranking of two flattened images are \( \rho = 0.137 \), 0.75 and 0.75 in (e) and (d) respectively. ranking scaled circle in (e) and (f) respectively.

Fig. 8: Image pixel ranking for each flattened window.

(a) \( \rho = 0.216 \)  
(b) \( \rho = 0.79 \)

(c) \( \rho = 0.137 \)  
(d) \( \rho = 0.75 \)

(e) \( \rho = -0.18 \)  
(f) \( \rho = 0.71 \)

Fig. 9: Sample of moving objects in (a) and (b) without and with averaging by window 2x2. The ranking of two flattened images are \( \rho = 0.216 \) and 0.79 in (a) and (b) respectively. Sample of moving noisy object in (c) and (d) without and with image averaging by a window size of 2x2. The ranking of two flattened images are \( \rho = 0.137 \), 0.75 and 0.75 in (e) and (d) respectively. ranking scaled circle in (e) and (f) respectively.

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 and reduces the size and scan till it reaches the highest correlation.

B. Feature Extraction

This paper proposes a new approach for feature selection based on data preference values by ranking the pixels instead of CNN convolution. The features are based on ranking computational space. Therefore, the kernel window size is considered a factor for feature selection.

1) Window Pixels' Ranking: For each scanned window in the image, the flatten ranked vector is ranked before measuring the \( \rho \) with the ranker kernel. The Fig. 8 shows the window size 3X3 range from \( \lambda_{k_1} = 23 \) to \( \lambda_{k_2} = 9 \).

Ranking the pixel reduce the data margin so it reduce the computational complexity.

2) Weighted Ranker Kernel: The kernel weights are randomly initialized from -0.05 to 0.05 learns the features by BP the weights. the partial change in the kernel is by differentiating the spearman correlation as in Eq. 4

\[
dKw = 2 \cdot Img - d\rho \cdot \frac{n^3 - n}{-6}
\]

Different kernel sizes could be used. However, we propose multiple kernels for big images’ size. We use three different kernels to capture the relations between different features in the image.

3) Max Pooling: We use the max. pooling approach to reduce the features map’s size and select the highest correlation values to feed to the PNN.

C. PN Structure

PN is the deep learning structure of PNN for image classification. It consists of five layers, ranking features map, 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 \( \nu = 2 \) for binary ranking to map the classification. The number of output neuron is the number of the classes. The structure is shown in Fig 10. PN have one or more ranker kernel 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 \( \rho(k, \pi_w) \) 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 [28]. Three kernels with sizes (3, 10, and 20) are used for CFAR-100 [29].
IV. ALGORITHMS

A. Baseline Algorithm

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

Algorithm 1: PNN learning flow

| Data: $D \in \{x_1, x_2, \ldots, x_d\}$ |
| Result: $\pi \in \{\lambda_{y_1}, \ldots, \lambda_{y_n}\}$ |
1. Randomly initialize weights $\omega_{i,j} \in (-0.05, 0.05)$
2. repeat
   \begin{enumerate}
   \item forall $\langle x_i, \pi_i \rangle \in D$ do
   \item $a_{i[k]} = \sum_{j=1}^{m} \varphi(a_{i \cdot \cdot \omega_{i,j}})$ // FF
   \item PNN BP()
   \item $\omega_{new} = \omega_{old} - \eta \cdot \delta_i$ //UW
   \end{enumerate}
3. until $\rho_{Avg} = 1$ or $\#\text{iterations} \geq 10^6$;

B. Ranking Visualization

PN 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 output values per iteration 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 number of nodes in the middle and output layers. $W_m$ and $W_o$ are weighted matrix and $SS_t$ is the activation function of number of instances $t$. The time complexity in Eq. 5

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

- BB starts with calculating the error of output layer $E_{ot} = \rho' \cdot \Delta o = E_{ot} \cdot SS' \cdot E_{ot} \cdot SS'$ and $E_{mt} = E_{mt} \cdot SS'$
Algorithm 2: PN Learning flow

1. Converting image to greyscale
2. Flattening image
3. Image pixel ranking
4. 2D Image
5. Pixel averaging by a 2X2 window
6. Flattening image
7. Select one/more kernel sizes.
8. Random init. Kernel
9. Flattening image
10. Compute $\rho(Img_w, Kw)$ feature map
11. Max. Pooling
12. Flattening image
13. PNN FF()
14. PNN BP()
15. PNN UW()
16. Max. Pooling BP()
17. Ranker kernel BP and UW()
18. repeat
19. 2D Image
20. Scanned window pixel ranking $Img_w$
21. Compute $\rho(Img_w, Kw)$ feature map
22. Max. Pooling
23. Flattening image
24. PNN FF()
25. PNN BP()
26. PNN UW()
27. Max. Pooling BP()
28. Ranker kernel BP and UW()
29. until $\rho_{Avg} = 1$ or $\#iterations \geq 10^6$;

Algorithm 3: PNN BP

1. Step 1: for each pn in Output layer do
2. $Err_i = \rho = -6 \cdot \frac{(\sum_{i} - x_i)}{m(m-1)}$ /spearman error
3. $\delta_i = Err \cdot \varphi$
2. Step 2: for each pn in middle layer do
3. $Err_i = \sum_{k=0}^{m} \omega_k \cdot \delta_k$
4. $\delta_i = Err \cdot \varphi$

then UW

$$W_m = W_m - Delta_m$$  \hspace{1cm} (6)

This time complexity is then multiplied by the number of epochs $n$

$$O(n \cdot m \cdot o \cdot t)$$  \hspace{1cm} (7)

2) Input Neurons: The number of PN input neurons is represented by Eq. 8

$$#Input = (Img_w - K_w + 1) \cdot (Img_H - K_H + 1)$$  \hspace{1cm} (8)

where $K_w$ is kernel width and $K_h$ is kernel height.

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.

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 neuron (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 has a stable rate of ranking convergence comparing to sigmoid, tanh, and ReLU. This stability is due to the stepstap width, which leads each point to reach the correct ranking during FF and BP in fewer epochs.

1) PSS and SS Evaluation: As shown in Fig 13, PSS reaches convergence and remains stable for a long number of iterations compared to SS. However, SS has better $\rho$ than PSS. This good performance of SS is due to the reason:

- The symmetry of SS function on the x axis. The SS shape handles both positive and negative normalized data. It reduces the number of iterations to reach the correct ranking values.

To have the same performance for SS and PSS, the input

Fig. 12: Visualizing the ranking of five labels using SS activation function of stock data set at epoch 0 and 3900 in (a) and (b) respectively.
Ranking all labels | 60% missing labels

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

data should be scaled from 0 to step width $X \#$steps and from -b to b for PSS and SS respectively.

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 iteration. 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.

4) **Dropout Regularization:** Dropout 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 updating weight steps. The comparison between dropout and non-dropout of PNN are shown in Fig. 15. The gap between the training model and ten-fold cross-validation curves has been reduced using dropout regularization of type A 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 [29] and Fashion-MNIST [30] data sets.

B. **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 exceptions preference mining [31], and the ‘algae’ data set is the first type that highlights the indifference preferences problem, where labels have repeated preference value [32]. 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 [11]. The third type of data set is semi-synthetic (SS) taken from the KEBI Data Repository at the Philips University of Marburg [27]. 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.

**TABLE II: Benchmark data sets for label ranking; preference mining [32], semi-synthetic (SS) [27] and real-world data sets**

<table>
<thead>
<tr>
<th>type</th>
<th>DS</th>
<th>category</th>
<th>#inst.</th>
<th>#attr.</th>
<th>#lbl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mining data</td>
<td>algae</td>
<td>chemical stat.</td>
<td>317</td>
<td>11</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>german.2005</td>
<td>user pref.</td>
<td>413</td>
<td>29</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>german.2009</td>
<td>user pref.</td>
<td>413</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>sushi</td>
<td>user pref.</td>
<td>5000</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>top?movies</td>
<td>user pref.</td>
<td>602</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Real data</td>
<td>cold</td>
<td>biology</td>
<td>2,465</td>
<td>23</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>diau</td>
<td>biology</td>
<td>2,465</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>dtt</td>
<td>biology</td>
<td>2,465</td>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>heat</td>
<td>biology</td>
<td>2,465</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>spo</td>
<td>biology</td>
<td>2,465</td>
<td>24</td>
<td>11</td>
</tr>
<tr>
<td>Semi-Synthesized data</td>
<td>authorship</td>
<td>A</td>
<td>841</td>
<td>70</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>bodyfat</td>
<td>B</td>
<td>252</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>calhousing</td>
<td>B</td>
<td>20,640</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>cpu-small</td>
<td>B</td>
<td>8192</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>elevators</td>
<td>B</td>
<td>16,599</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>fried</td>
<td>B</td>
<td>40,769</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>glass</td>
<td>A</td>
<td>214</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>housing</td>
<td>B</td>
<td>506</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>iris</td>
<td>A</td>
<td>150</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>pendigits</td>
<td>A</td>
<td>10,992</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>segment</td>
<td>A</td>
<td>2310</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>stock</td>
<td>B</td>
<td>950</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>vehicle</td>
<td>A</td>
<td>846</td>
<td>18</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>vowel</td>
<td>A</td>
<td>528</td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>wine</td>
<td>A</td>
<td>178</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>wisconsin</td>
<td>B</td>
<td>194</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

B. **Results**

1) **Image Classification Results:** PN has 3 kernel sizes of 5,10 and 20 and is tested on the CFAR-100 [29] data set and 1 kernel with a size 5 for Fashion-MNIST data set [30]. Table III 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 Kendal $\tau$ coefficient for comparison with other approaches. For data validation, we used with 10-fold cross-validation. To avoid the over-fitting problem, We used hyperparameters, i.e. l.r. = (0.005, 0.05, 0.1) hidden neuron = no.inputs+(5, 10, 50, 100, 200) 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 IV 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 [36], supervised decision tree [27], MLP label ranking [16], and label ranking tree forest (LRT) [37]. 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. All the results are held using a single hidden layer with various hidden neurons (50 to 300) and SS activation function. The Kendall $\tau$ error converges and reaches close to 1 after 2000 iterations, as shown in Fig. 16.

Table IV compares PNN with the similar approaches used for label ranking. These approaches are; Decision trees [36], MLP-LR [16] and label ranking trees forest LRT [37]. In this comparison, we choose the method that has the best results for each approach.

![ROC for ranking 3 labels](image)

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

![Without dropout](image) ![With dropout](image)

Fig. 15: Training and validation performance without and with dropout regulation approach in (a) and (b) respectively.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Model</th>
<th>Baseline</th>
<th>MixUp</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-100</td>
<td>ResNet [33]</td>
<td>72.22</td>
<td>78.9</td>
</tr>
<tr>
<td></td>
<td>WRN [34]</td>
<td>78.26</td>
<td>82.5</td>
</tr>
<tr>
<td></td>
<td>Dense [35]</td>
<td>81.73</td>
<td>83.23</td>
</tr>
<tr>
<td></td>
<td>PrefNet</td>
<td>80.6</td>
<td>-</td>
</tr>
<tr>
<td>Fashion-MNIST</td>
<td>MLP</td>
<td>0.871</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>RandomForest</td>
<td>0.873</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>LogisticRegression</td>
<td>0.842</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SVC</td>
<td>0.897</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SGDClassifier</td>
<td>0.81</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>PrefNet</td>
<td>0.91</td>
<td>-</td>
</tr>
</tbody>
</table>

TABLE IV: PNN performance comparison with various approaches: supervised clustering [36], supervised decision tree [27], MLP label ranking [16] and label ranking tree forest (LRT) [37]

<table>
<thead>
<tr>
<th>Preference Mining Data</th>
<th>Avg. τ</th>
<th>Lstep</th>
<th>#m.n.</th>
</tr>
</thead>
<tbody>
<tr>
<td>algae</td>
<td>0.751</td>
<td>0.005</td>
<td>100</td>
</tr>
<tr>
<td>german2005</td>
<td>0.89</td>
<td>0.005</td>
<td>20</td>
</tr>
<tr>
<td>german2009</td>
<td>0.78</td>
<td>0.005</td>
<td>20</td>
</tr>
<tr>
<td>sushi</td>
<td>0.69</td>
<td>0.005</td>
<td>300</td>
</tr>
<tr>
<td>top 7 movies</td>
<td>0.602</td>
<td>0.005</td>
<td>20</td>
</tr>
</tbody>
</table>

4) Preference Mining Results: Ranking performance of the new preference mining data set is represented in Table II. 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.

Experiments on the biological real-world data set were conducted using supervised clustering (SC) [36], Table V presents the comparison between PNN type A and supervised clustering on biological real world data in terms of \( \text{Loss}_{LR} \) as given in Eq. 9.

\[
\tau = 1 - 2 \cdot \text{Loss}_{LR} 
\]

where \( \tau \) is Kendall \( \tau \) ranking error and \( \text{Loss}_{LR} \) is the ranking loss function.
TABLE VI: Comparison between PNN type A and supervised clustered on biological real world data in terms of LossLR

<table>
<thead>
<tr>
<th>Biological real world data</th>
<th>S.Clustering</th>
<th>PNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>cold</td>
<td>0.198</td>
<td>0.11</td>
</tr>
<tr>
<td>dian</td>
<td>0.304</td>
<td>0.255</td>
</tr>
<tr>
<td>dtt</td>
<td>0.124</td>
<td>0.013</td>
</tr>
<tr>
<td>heat</td>
<td>0.072</td>
<td>0.014</td>
</tr>
<tr>
<td>spo</td>
<td>0.118</td>
<td>0.0804</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.1652</strong></td>
<td><strong>0.0804</strong></td>
</tr>
</tbody>
</table>

C. Computational Platform

PNN and PN are 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 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 non-GPU version of PNN is located in Github Repository [38].

D. Discussion and Future Work

It can be noticed from table II that PN is performing better than ResNet [33] and WRN [34]. Different types of architectures of PN could be used to enhance the results and reach state-of-the-art in terms of image classification. 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 LossLR 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 rank 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 $(\lambda_a > \lambda_b, \lambda_b > \lambda_c)$ is encoded to $(1, 2, -1)$ and $(\lambda_a > \lambda_b, \lambda_c > \lambda_d)$ 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 $(\lambda_a > \lambda_b \geq \lambda_c)$ is encoded to $(1, 2, 3)$ or $(1, 2, 2)$. Future research may focus on modifying PNN architecture by adding bias and solving problems of extreme multi-label ranking.

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 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, and activation functions are modeled using wolframe Mathematica software [39]. A video demo that shows the ranking learning process using toy data is available to download [23].

ACKNOWLEDGEMENT

This work was supported by the Australian Research Council (ARC) under discovery grants DP180100670 and DP180100656.

REFERENCES

TABLE VII: PNN ranking performance in terms of $r$ coefficient, learning step and number of hidden neurons.

<table>
<thead>
<tr>
<th>type</th>
<th>Real Data</th>
<th></th>
<th></th>
<th></th>
<th>Semi-Synthesized Data</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DS</td>
<td>Avg. $r$</td>
<td>#m.n.</td>
<td>#Iterations</td>
<td>Scaling</td>
<td>Avg. $r$</td>
<td>#m.n.</td>
</tr>
<tr>
<td>cold</td>
<td>0.4</td>
<td>14</td>
<td>2000</td>
<td>-10-10</td>
<td>-40-40</td>
<td>0.87</td>
<td>100</td>
</tr>
<tr>
<td>diax</td>
<td>0.466</td>
<td>20</td>
<td>2000</td>
<td>-10-10</td>
<td>-40-40</td>
<td>0.87</td>
<td>100</td>
</tr>
<tr>
<td>dtt</td>
<td>0.60</td>
<td>10</td>
<td>2000</td>
<td>-10-10</td>
<td>-40-40</td>
<td>0.87</td>
<td>100</td>
</tr>
<tr>
<td>heat</td>
<td>0.876</td>
<td>10</td>
<td>2000</td>
<td>-10-10</td>
<td>-40-40</td>
<td>0.87</td>
<td>100</td>
</tr>
<tr>
<td>spo</td>
<td>0.8</td>
<td>20</td>
<td>2000</td>
<td>-10-10</td>
<td>-40-40</td>
<td>0.87</td>
<td>100</td>
</tr>
<tr>
<td>German2005</td>
<td>0.8</td>
<td>20</td>
<td>2000</td>
<td>-10-10</td>
<td>-40-40</td>
<td>0.87</td>
<td>100</td>
</tr>
<tr>
<td>German2009</td>
<td>0.87</td>
<td>20</td>
<td>2000</td>
<td>-10-10</td>
<td>-40-40</td>
<td>0.87</td>
<td>100</td>
</tr>
<tr>
<td>authorship</td>
<td>0.915</td>
<td>30</td>
<td>100000</td>
<td>-20-200</td>
<td>-40-40</td>
<td>0.90</td>
<td>100</td>
</tr>
<tr>
<td>bodyfat</td>
<td>0.559</td>
<td>100</td>
<td>100000</td>
<td>-20-200</td>
<td>-40-40</td>
<td>0.90</td>
<td>100</td>
</tr>
<tr>
<td>calhousing</td>
<td>0.34</td>
<td>20</td>
<td>50000</td>
<td>-40-40</td>
<td>-40-40</td>
<td>0.90</td>
<td>100</td>
</tr>
<tr>
<td>cpu-small</td>
<td>0.46</td>
<td>50</td>
<td>50000</td>
<td>-60-60</td>
<td>-40-40</td>
<td>0.90</td>
<td>100</td>
</tr>
<tr>
<td>elevators</td>
<td>0.13</td>
<td>20</td>
<td>50000</td>
<td>-40-40</td>
<td>-40-40</td>
<td>0.90</td>
<td>100</td>
</tr>
<tr>
<td>fraud</td>
<td>0.89</td>
<td>100</td>
<td>10000</td>
<td>-20-200</td>
<td>-40-40</td>
<td>0.90</td>
<td>100</td>
</tr>
<tr>
<td>glass</td>
<td>0.91</td>
<td>100</td>
<td>10000</td>
<td>-20-200</td>
<td>-40-40</td>
<td>0.90</td>
<td>100</td>
</tr>
<tr>
<td>housing</td>
<td>0.712</td>
<td>25</td>
<td>50000</td>
<td>-20-20</td>
<td>-40-40</td>
<td>0.90</td>
<td>100</td>
</tr>
<tr>
<td>iris</td>
<td>0.917</td>
<td>100</td>
<td>50000</td>
<td>-1-1</td>
<td>-40-40</td>
<td>0.90</td>
<td>100</td>
</tr>
<tr>
<td>pdigits</td>
<td>0.86</td>
<td>100</td>
<td>50000</td>
<td>-10-10</td>
<td>-40-40</td>
<td>0.90</td>
<td>100</td>
</tr>
<tr>
<td>segment</td>
<td>0.916</td>
<td>20</td>
<td>50000</td>
<td>-20-20</td>
<td>-40-40</td>
<td>0.90</td>
<td>100</td>
</tr>
<tr>
<td>stock</td>
<td>0.834</td>
<td>50</td>
<td>50000</td>
<td>-60-60</td>
<td>-40-40</td>
<td>0.90</td>
<td>100</td>
</tr>
<tr>
<td>vehicle</td>
<td>0.754</td>
<td>100</td>
<td>10000</td>
<td>-100-100</td>
<td>-40-40</td>
<td>0.90</td>
<td>100</td>
</tr>
<tr>
<td>vowel</td>
<td>0.85</td>
<td>22</td>
<td>10000</td>
<td>-50-50</td>
<td>-40-40</td>
<td>0.90</td>
<td>100</td>
</tr>
<tr>
<td>wine</td>
<td>0.90</td>
<td>100</td>
<td>50000</td>
<td>-60-60</td>
<td>-40-40</td>
<td>0.90</td>
<td>100</td>
</tr>
<tr>
<td>wisconsin</td>
<td>0.81</td>
<td>100</td>
<td>10000</td>
<td>-60-60</td>
<td>-40-40</td>
<td>0.90</td>
<td>100</td>
</tr>
</tbody>
</table>