Preference Neural Network

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

***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. *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, 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 $\mathbb{X}$, there is a label $\mathcal{L}$ associated with x, $\mathcal{L} \in \pi$, where $\pi = \{\lambda_1, .., \lambda_n\}$, 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 $\pi = \{1, 2, \cdots, n\}$. The label order has the following three features. irreflexive where $\lambda_a \nsucc \lambda_a$,transitive where $(\lambda_a \succ \lambda_b) \wedge (\lambda_b \succ \lambda_c) \implies \lambda_a \succ \lambda_c$ and asymmetric $\lambda_a \succ \lambda_b \implies \lambda_b \nsucc \lambda_a$. 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 $\pi = (1, 2, 2, 3)$, where a, b, c, and, d are the label indexes and $\lambda_a, \lambda_b, \lambda_c$ and λ_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 λ_c is not comparable to $(\lambda_a, \lambda_b, \lambda_d)$.

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 $\lambda_i \succ \lambda_j$ or $\lambda_j \prec \lambda_i$ for an input x. Statistical methods includes 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 value of rectified linear unit (*Relu*), *Sigmoid*, or *Softmax* activation functions. Predictive probability can be enhanced by changing the function slope to be almost a step shape to create multiple almost discrete values.
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 $\pi = (\lambda_a \succ \lambda_b \succ \lambda_c)$ and $\mathcal{D} = (1,1,1)$ for $\pi = (\lambda_a \succ \lambda_c \succ \lambda_b)$ the ranking is unique; however, pairwise classification creates no ground truth ranking for the pair $\lambda_b \succ \lambda_c$ and $\lambda_c \succ \lambda_b$ 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) *PN* 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 τ 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 τ 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 ρ will be discussed in section IV. The ranking convergence reaches to $\tau \simeq 1$ 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 τ for stopping criteria. Kendall τ 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 τ and *RMS* and *PNN* ranking evaluation using ρ and *RMS*.
2) The convergence performance takes numbers of iterations to reach the ranking $\tau \simeq 1$ 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 $\mathcal{L} = \{\lambda_{y_1}, \ldots, \lambda_{y_n}\}$. *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 *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 ∞. 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 represents the preference values from {*0* to n} where 0 represent the incomparable relation $\perp$ 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

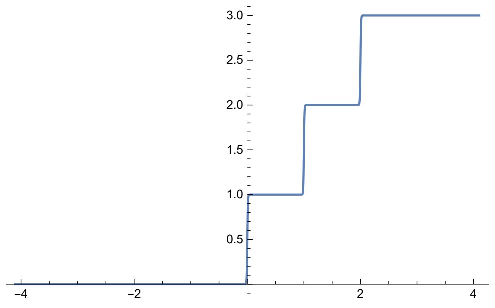
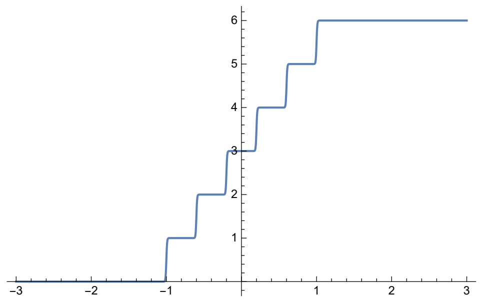


Fig. 1: *PSS* activation function where $n = 3$ and step width $w = 1$

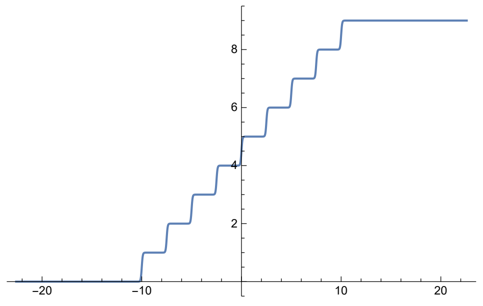
$$y = -\frac{1}{2}\Big(\sum_{i=0}^{n}\tanh(-100(x - wi))\Big) + \frac{n}{2} \tag{1}$$

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.



(a)



(b)

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.

$$y = -\frac{1}{2}\Big(\sum_{i=0}^{n}\tanh(\frac{-100}{b}x + c(1 - \frac{2i}{n-1}))\Big) + \frac{n}{2} \tag{2}$$

where c = 100, n = 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 ∞ 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 Therefore, input values must be scaled from -b to b. The step width is 1 when n$\simeq 2b$. 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 τ [25] and *spearman* ρ [26]. However, the Kendall τ function lacks continuity and differentiability. Therefore, The *spearman* ρ correlation coefficient is used to measure the ranking between output labels. *spearman* ρ error derivative is used as a gradient ascent process for *BP*, and correlation is used as a ranking evaluation function for convergence stopping criteria. τ_{Avg} is the average τ per label divided by the number of instances m, as shown in line 8 of Algorithm 1. *spearman* ρ 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 $\tau = 0.3$.

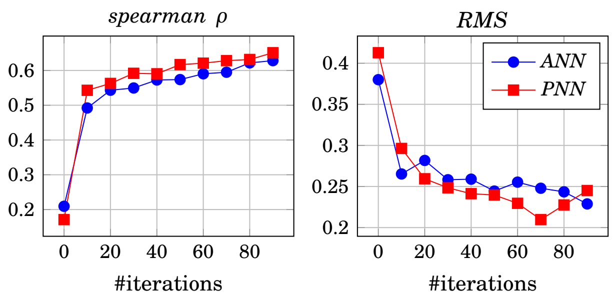


Fig. 3: Ranker network and *PNN* evaluation in terms of *RMS* and *spearman* correlation error functions

Fig 3 shows the comparison between the initial ranker network and *PNN*; the ranker network uses Kendall τ in 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 Eq.3

$$\rho = 1 - \frac{6\sum_{i=1}^{m}(y_i - yt_i)^2}{m(m^2-1)} \tag{3}$$

where y_i, yt_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 $\rho \simeq 1$. 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 arrows links that represent the multi-deterministic values. The *PN* in the middle layer connects to only n output neurons $stp = n+1$; where stp is the number of *SS* steps. The *PN* in output layer represents the preference value. The middle and output *PN*s produce a preference value from 0 to ∞ as illustrated in Fig. 5.

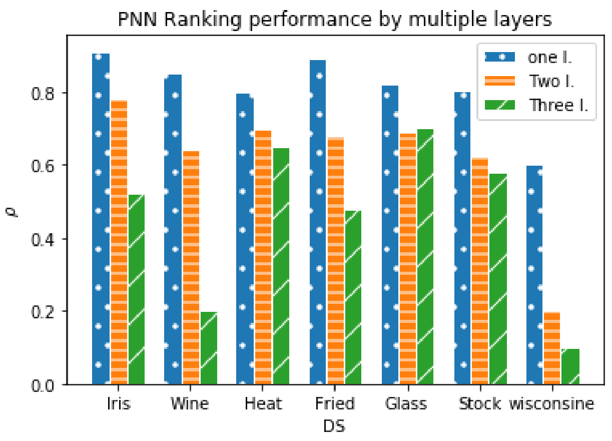


Fig. 4: Multiple layer label ranking comparison of benchmark data sets [27] results using the *PNN* and *SS* functions after 100 epochs and learning rate = 0.007.

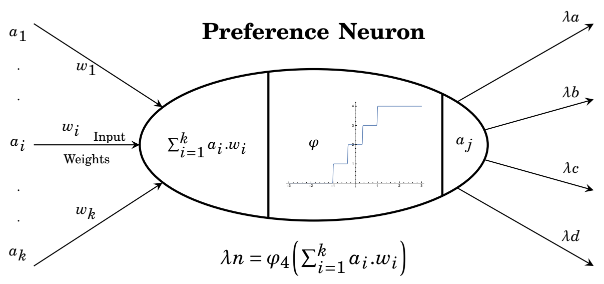


Fig. 5: The structure of preference neuron where $\varphi_{n=4}$.

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, $H_n \geq \mathcal{L}_n$, 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 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 $\rho \simeq 1$ 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

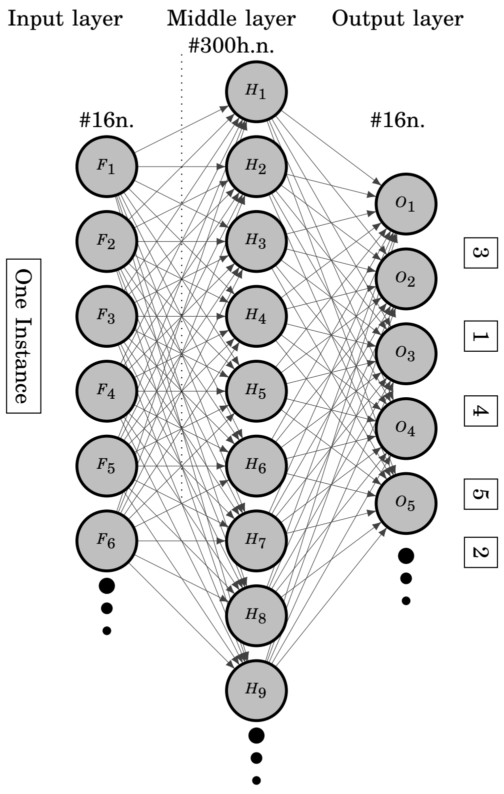


Fig. 6: *PNN* where $\varphi_{n=16}$, $f_{in} = 16$ and $\lambda_{out} = 16$, per $\langle x_1, \pi_1 \rangle$, $\mathcal{L} \in \{\lambda_a, \lambda_b, \lambda_c, \lambda_d\}$ where $\pi_1 = \{1,2,3,4,\ldots,16\}$.

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 $\rho_{Avg} \simeq 1$ 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 $\simeq 1$ 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*.

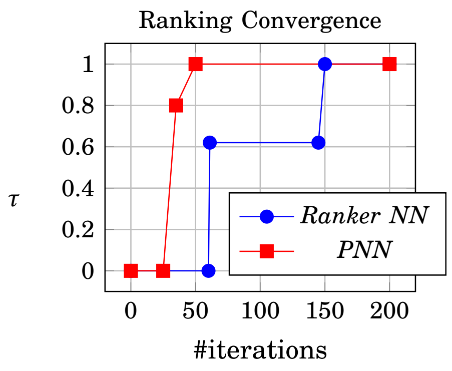


Fig. 7: The structure used in both ranker *ANN* and *PNN* where $\varphi_{n=3}$, $f_{in} = 3$ and $\lambda_{out} = 3$, per $\langle x_1, \pi_1 \rangle$, $\mathcal{L} \in \{\lambda_a, \lambda_b, \lambda_c\}$ where $\pi_1 = \{1,2,3\}$. and comparison of the convergence for both NN's. The demo video of convergence of two NN in the link [23].

TABLE I: *ANN* types used in initial experiment.

Type	**Ranker *ANN***	***PNN***
Activation Fun.	*ReLU,Sigmoid*	PSS, SS
Gradient	Descent	Ascent
Objective Fun.	*RMS*	ρ
Stopping Criteria.	τ	ρ

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 $\pi = \{\lambda_1, .., \lambda_m\}$ to $\pi = \{\lambda_1, .., \lambda_k\}$ where the maximum greyscale value $\lambda_m = 255$ and λ_k is the maximum ranked pixel value as illustrated in Fig. 8

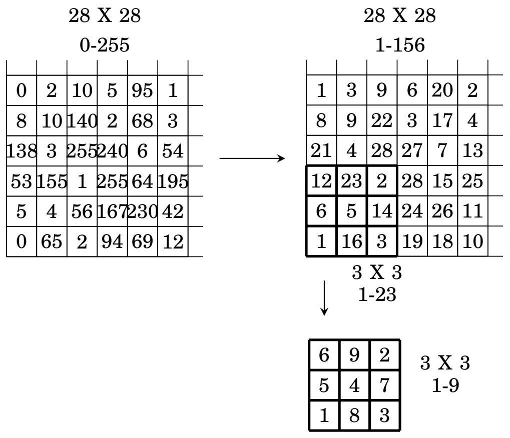


Fig. 8: Image pixel ranking for each flattened window.

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 ρ 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).

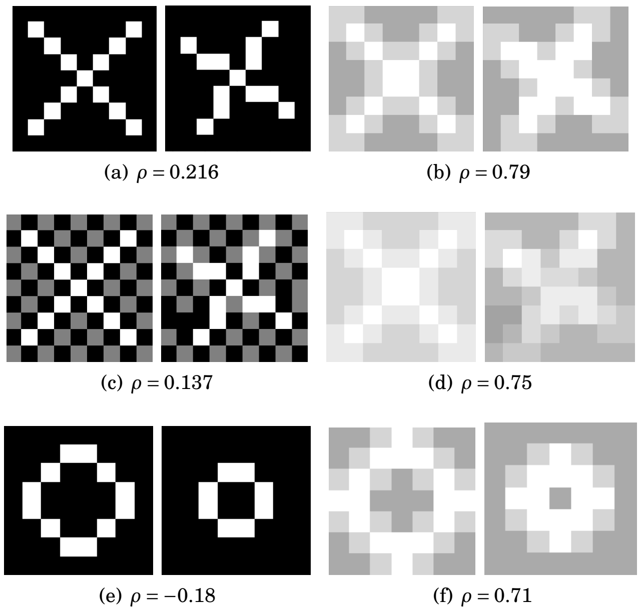


Fig. 9: Sample of moving objects in (a) and (b) without and with averaging by window 2x2. The ranking of two flattened images are ρ = 0.216 and 0.79 in (a) and (b) respectively. Sample of moving noisy object in (c) and (d) without and with image averaging by a window of 2x2. The ranking of two flattened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

The two approaches Pixel ranking and Averaging has been applied on 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 and reduces the size and scan till it reaches the highest correlation.

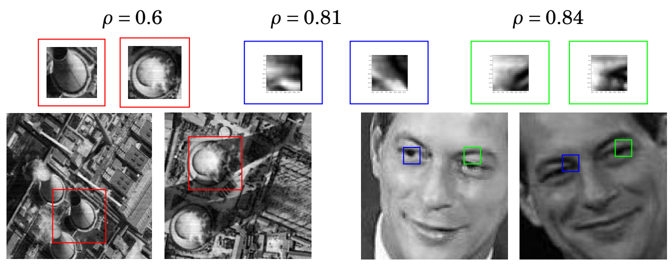


Fig. 10: Detecting the similarity in remote sensing and face recognition by ranking the image pixels after averaging the pixels using a 2x2 window.

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 ρ 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

$$dKw = 2 \cdot Img - d\rho \cdot \frac{n^3 - n}{-6} \tag{4}$$

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 *PNN*s 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 neuron is the number of the classes. The structure is shown in Fig 11. *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 $\rho(\pi_k, \pi_w)$ where π_k is the kernel preference values and π_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.

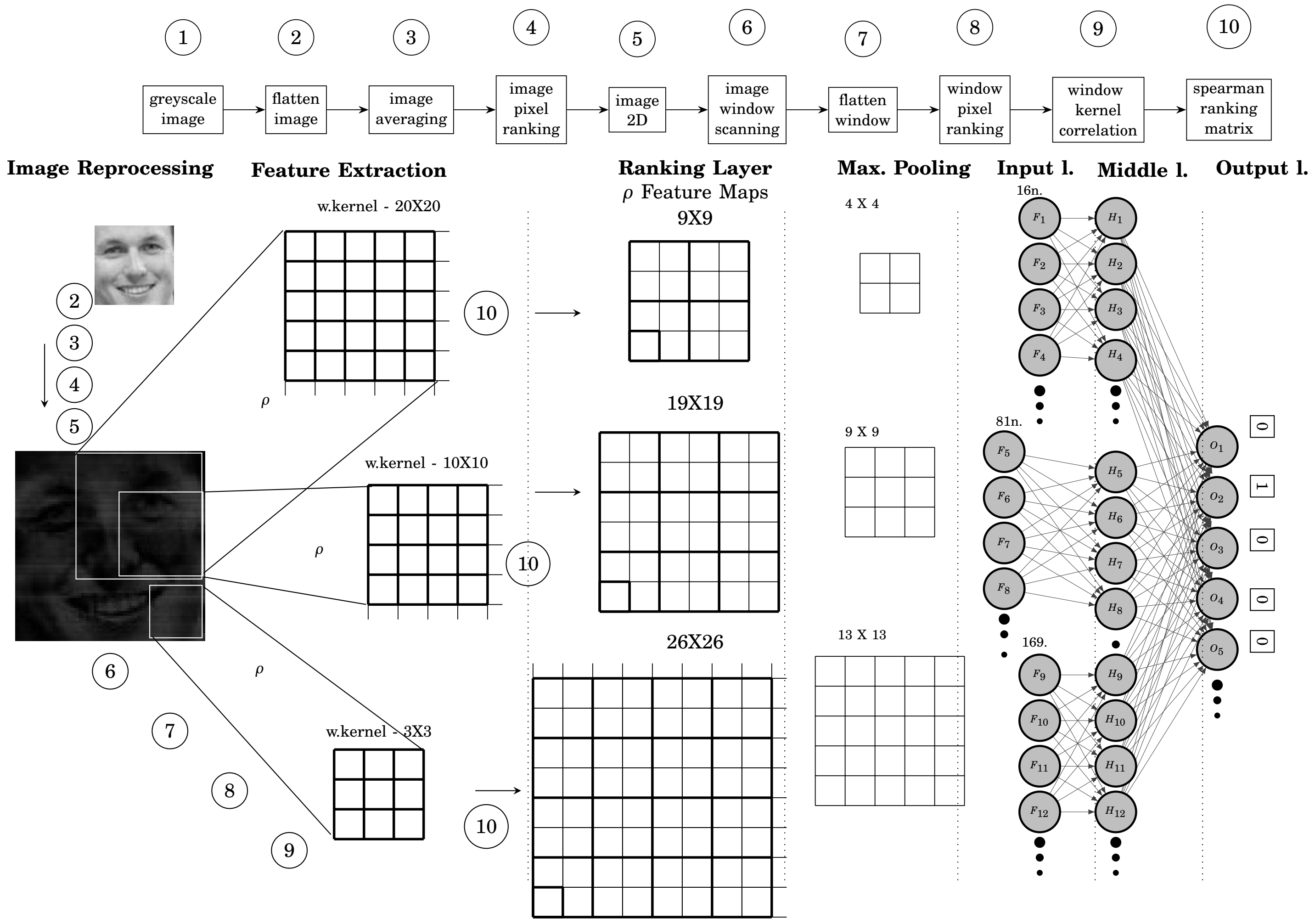


Fig. 11: The *PN* structure has three kernels and three *PNN*s where $\varphi_{n=2}$, $f_{1in}=16, f_{2in}=81, f_{3in}=169$ and $\lambda_{out}=15$, per $\langle x_1,\pi_1\rangle, \pi\in\{\lambda_1,\lambda_2,\lambda_3\cdots,\lambda_{15}\}$.

Algorithm 1: *PNN* learning flow

```
  Data: $\mathcal{D}\in\{x_1,x_2,\dots,x_d\}$
  Result: $\pi\in\{\lambda_{y_1},\dots,\lambda_{y_n}\}$
1 Randomly initialize weights $\omega_{i,j}\in\{-0.05,0.05\}$
2 repeat
3     forall $\langle x_i,\pi_i\rangle\in\mathcal{D}$ do
4         $a_i|_{l-1}=\sum_{i=1}^{m}\varphi(a_i\cdot\omega_i)|_n$ // FF
5         PNN BP()
6         $\omega_{i_{new}}=\omega_{i_{old}}-\eta\cdot\delta_i$ //UW
7 until $\rho_{Avg}$ = 1 or #iterations $\geq 10^6$;
```

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* 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 τ 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

$$\mathcal{O}(m\cdot o\cdot t) \tag{5}$$

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

$$W_m=W_m-Delta_m \tag{6}$$

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

$$\mathcal{O}(n\cdot m\cdot o\cdot t) \tag{7}$$

Algorithm 2: *PN* Learning flow

```
8  Converting image to greyscale
9  Flattening image
10 Image pixel ranking
11 2D Image
12 Pixel averaging by a 2X2 window
13 Flattening image
14 Select one/more kernel sizes.
15 Random init. Kernel $K\omega_{x,y} \in \{-0.05, 0.05\}$
16 Random init. PNN $\omega_{i,j} \in \{-0.05, 0.05\}$
17 repeat
18     2D Image
19     Scanned window pixel ranking $Img_w$
20     Compute $\rho(Img_w, Kw)$ feature map
21     Max. Pooling.
22     Flattening image
23     PNN FF()
24     PNN BP()
25     PNN UW()
26     Max. Pooling BP()
27     Ranker kernel BP and UW()
28 until $\rho_{Avg}$ = 1 or #iterations $\geq 10^6$;
```

Algorithm 3: *PNN* BP

```
29 Step 1: for each $pn_i$ in Output layer do
30     $Err_i = \rho = -6 \cdot \frac{(2yt_i - y_i)}{m(m^2-1)}$ //spearman error
31     $\delta_i = Err \cdot \varphi\prime$
32 Step 2: for each $pn_i$ in middle layer do
33     $Err_i = \sum_{k=0}^{m} \omega_k \cdot \delta_k$
34     $\delta_i = Err \cdot \varphi\prime$
```

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

$$\#Input = (Img_W - K_W + 1) \cdot (Img_H - K_H + 1) \quad (8)$$

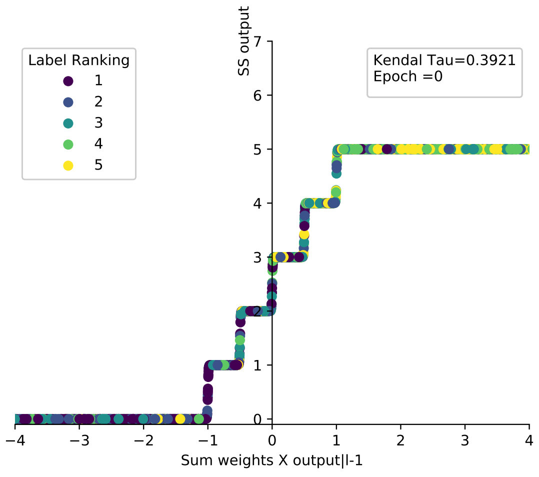
where K_W is kernel width and K_H is kernel height.

V. Network Evaluation

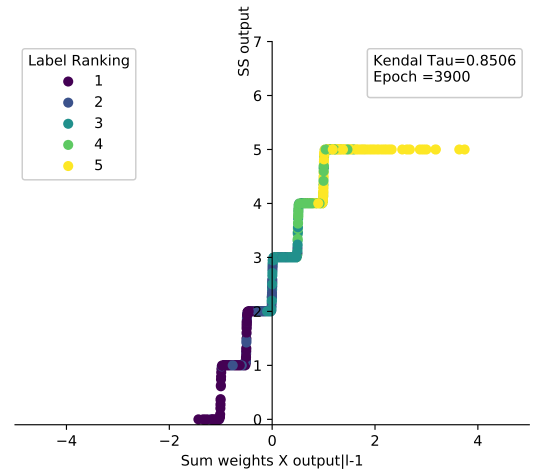
This section evaluates the *PNN* against different activation functions and architectures. All weights are initialized = 0 to compare activation functions and *A* and *B* have the same initialized random weights to evaluate the structure.

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 stairstep width, which leads each point to reach the correct ranking during *FF* and *BP* in fewer epochs.



(a)



(b)

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.

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 ρ 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 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

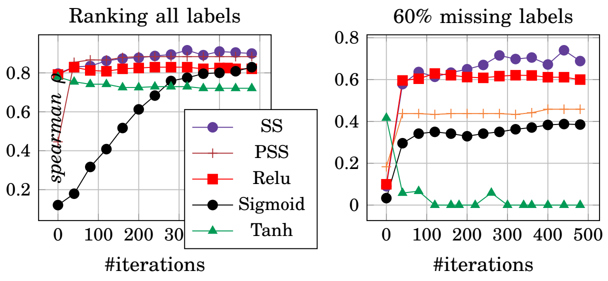


Fig. 13: *PNN* activation function comparison using complete labels and 60% missing labels in (a) and (b), respectively.

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 as shown in Fig. 14

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.

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 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 Philipps 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

type	DS	category	#inst.	#attr.	#lbl.
Mining data	algae	chemical stat.	317	11	4
	german.2005	user pref.	413	29	5
	german.2009	user pref.	413	32	5
	sushi	user pref.	5000	10	10
	top7movies	user pref.	602	7	7
Real data	cold	biology	2,465	23	4
	diau	biology	2,465	24	6
	dtt	biology	2,465	24	4
	heat	biology	2,465	24	6
	spo	biology	2,465	24	11
Semi-Synthesized data	authorship	A	841	70	4
	bodyfat	B	252	7	7
	calhousing	B	20,640	6	5
	cpu-small	B	8192	3	4
	elevators	B	16,599	9	9
	fried	B	40,769	9	5
	glass	A	214	9	6
	housing	B	506	6	6
	iris	A	150	4	3
	pendigits	A	10,992	16	10
	segment	A	2310	3	4
	stock	B	950	5	5
	vehicle	A	846	18	4
	vowel	A	528	10	11
	wine	A	178	13	3
	wisconsin	B	194	16	16

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.

TABLE III: Comparison of classification on CIFAR-100 [29] and Fashion-Mnist data set [30] Data sets using different convolution models

DS	Model	Baseline	MixUp
CIFAR-100	ResNet [33]	72.22	78.9
	WRN [34]	78.26	82.5
	Dense [35]	81.73	83.23
	PrefNet	80.6	-
Fashion-MNIST	MLP	0.871	-
	RandomForest	0.873	-
	LogisticRegression	0.842	-
	SVC	0.897	-
	SGDClassifier	0.81	-
	PrefNet	0.91	-

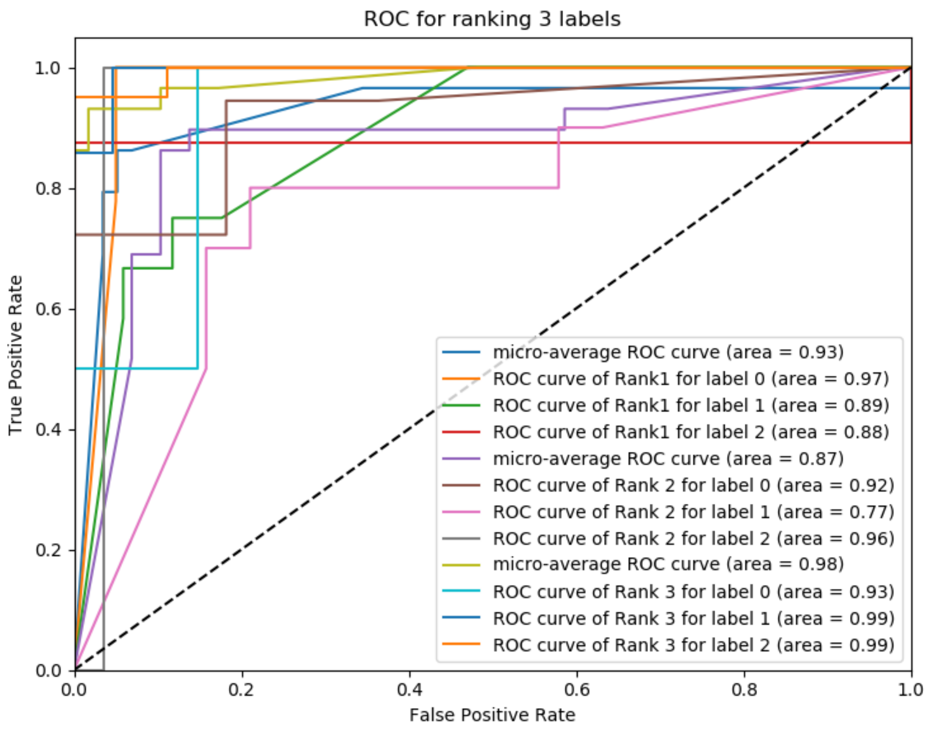


Fig. 14: ROC of three label ranking on the wine data set using *PNN* h.n=100 and 50 epochs

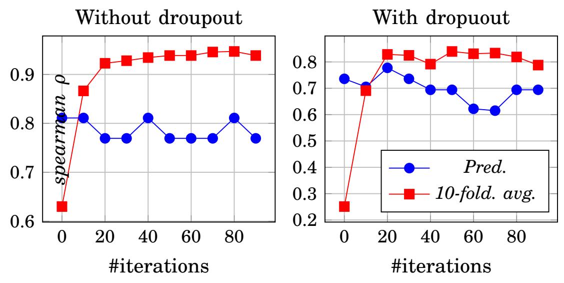


Fig. 15: Training and validation performance without and with dropout regulation approach in (a) and (b) respectively.

2) Label Ranking Results: PNN is evaluated by restricted and non-restricted label ranking data sets. The results are derived using *spearman* ρ and converted to *Kendall* τ 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 τ 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 τ 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.

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.

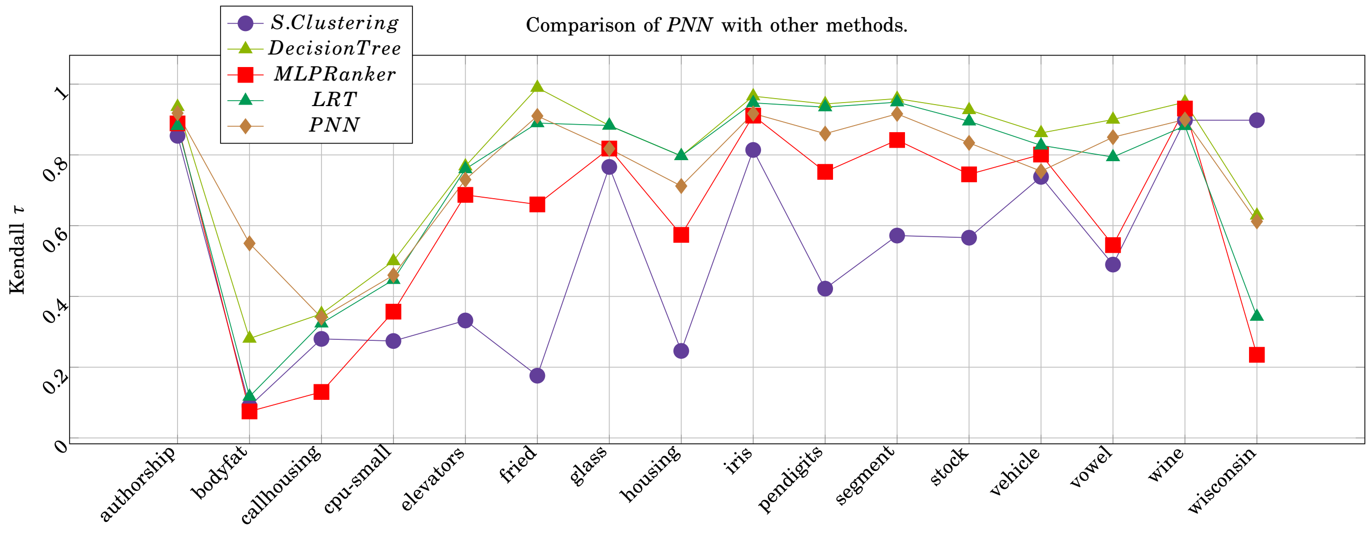


Fig. 16: Ranking performance comparison of *PNN* with other approaches.

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]

Label Ranking Methods					
DS	**S.Clust.**	***DT***	***MLP-LR***	***LRT***	***PNN***
authorship	0.854	0.936(IBLR)	0.889(LA)	0.882	0.918
bodyfat	0.09	0.281(CC)	0.075(CA)	0.117	0.5591
calhousing	0.28	0.351(IBLR)	0.130(SSGA)	0.324	0.34
cpu-small	0.274	0.50(IBLR)	0.357(CA)	0.447	0.46
elevators	0.332	0.768(CC)	0.687(LA)	0.760	0.73
fried	0.176	0.99(CC)	0.660(CA)	0.890	0.91
glass	0.766	0.883(LRT)	0.818(LA)	0.883	0.8175
housing	0.246	0.797(LRT)	0.574(CA)	0.797	0.712
iris	0.814	0.966(IBLR)	0.911(LA)	0.947	0.917
pendigits	0.422	0.944(IBLR)	0.752(CA)	0.935	0.86
segment	0.572	0.959(IBLR)	0.842(CA)	0.949	0.916
stock	0.566	0.927(IBLR)	0.745(CA)	0.895	0.834
vehicle	0.738	0.862(IBLR)	0.801(LA)	0.827	0.754
vowel	0.49	0.90(IBLR)	0.545(CA)	0.794	0.85
wine	0.898	0.949(IBLR)	0.931(LA)	0.882	0.90
wisconsin	0.09	0.629(CC)	0.235(CA)	0.343	0.612
Average	0.475	0.79	0.621	0.730	0.755

TABLE V: Preference mining ranking performance in terms of the Kendall τ coefficient and learning step and number of hidden neurons.

Preference Mining Data			
DS	**Avg.τ**	**l.step**	**#m.n.**
algae	0.751	0.005	100
german2005	0.89	0.005	20
german2009	0.78	0.005	20
sushi	0.69	0.005	300
top7 movies	0.602	0.005	20

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 $Loss_{LR}$ as given in Eq. 9.

$$\tau = 1 - 2 \cdot Loss_{LR} \tag{9}$$

where τ is Kendall τ 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. In order to enhance ranking performance, the data set has many labels. The number of hidden neurons is increased in order to exceed $\tau = 0.5$.

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

TABLE VI: Comparison between *PNN* type *A* and supervised clustered on biological real world data in terms of $Loss_{LR}$

Biological real world data		
DS	**S.Clustering**	***PNN***
cold	0.198	0.11
diau	0.304	0.255
dtt	0.124	0.01
heat	0.072	0.013
spo	0.118	0.014
Average	0.1632	0.0804

Computing cluster based on Linux RedHat 7.7, which has an NVIDIA Quadro GV100 and memory of 32 G.B.

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 $Loss_{LR}$ 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 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 $(\lambda_a \succ \lambda_b \perp \lambda_c)$ is encoded to (1, 2, -1) and $(\lambda_a \succ \lambda_b) \perp (\lambda_c \succ \lambda_d)$ 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 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 [38]. 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

[1] J. Frnkranz and E. Hllermeier, *Preference Learning*, 1st ed. Berlin, Heidelberg: Springer-Verlag, 2010.

[2] R. Brafman and C. Domshlak, “Preference handling - an introductory tutorial,” 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,” pp. 734–749, 2005.

[4] M. Montaner and B. López, “A taxonomy of recommender agents on the internet.” pp. 285–330, 2003.

[5] F. Aiolli, “A preference model for structured supervised learning tasks,” pp. 557–560, 2005.

[6] K. Crammer and Y. Singer, “Pranking with ranking,” pp. 641–647, 2002.

[7] Y. Zhou, Y. Liu, J. Yang, X. He, and L. Liu, “A taxonomy of label ranking algorithms,” *JCP*, vol. 9, pp. 557–565, 2014.

[8] J. Furnkranz and E. Hüllermeier, “Pairwise preference learning and ranking in machine learning,” pp. 145–156, 2003.

[9] J. Fürnkranz and E. Hüllermeier, “Preference learning,” 2010.

[10] S. Har-Peled, D. Roth, and D. Zimak, “Constraint classification: A new approach to multiclass classification,” 2002.

[11] E. Hüllermeier, J. Furnkranz, W. Cheng, and K. Brinker, “Label ranking by learning pairwise preferences,” pp. 1897–1916, 2008.

[12] J. Furnkranz and E. Hüllermeier, “Decision tree modeling for ranking data,” pp. 83–106, 2011.

[13] W. Cheng and E. Hüllermeier, “Instance-based label ranking using the mallows model,” pp. 143–157, 2008.

[14] G. Mihajlo, D. Nemanja, and V. Slobodan, “Learning from pairwise preference data using gaussian mixture model,” 2014.

[15] T. S. Chris Burges, “Learning to rank using gradient descent,” pp. 58–86, 2005.

[16] G. Ribeiro, W. Duivesteijn, C. Soares, and A. Knobbe, “Multilayer perceptron for label ranking,” in *Proceedings of the 22nd International Conference on Artificial Neural Networks and Machine Learning - Volume Part II*, ser. ICANN'12. Springer, 2012, p. 25–32.

[17] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, “An efficient boosting algorithm for combining preferences,” *J. Mach. Learn. Res.*, vol. 4, no. null, p. 933–969, Dec. 2003.

[18] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao, “Adapting boosting for information retrieval measures,” vol. 13, no. 3, p. 254–270, Jun. 2010. [Online]. Available: https://doi.org/10.1007/s10791-009-9112-1

[19] Y. Jian, J. Xiao, Y. Cao, A. Khan, and J. Zhu, “Deep pairwise ranking with multi-label information for cross-modal retrieval,” in *2019 IEEE International Conference on Multimedia and Expo (ICME)*, 2019, pp. 1810–1815.

[20] J. Li, W. Y. Ng Wing, T. Xing, S. Kwong, and H. Wang, “Weighted multi-deep ranking supervised hashing for efficient image retrieval,” *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 4, pp. 883–897, 2020, copyright - 2019© Springer-Verlag GmbH Germany, part of Springer Nature 2019; Last updated - 2020-03-08. [Online]. Available: http://ezproxy.lib.uts.edu.au/login?url=https://www-proquest-com.ezproxy.lib.uts.edu.au/scholarly-journals/weighted-multi-deep-ranking-supervised-hashing/docview/2373640699/se-2?accountid=17095

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

type	**DS**	**Avg. τ**	**#m.n.**	**#Iterations.**	**Scaling.**
Real Data	cold	0.4	10	2000	-40:40
	diau	0.466	20	2000	-20:20
	dtt	0.60	10	2000	-10:10
	heat	0.876	10	2000	-10:10
	spo	0.8	20	2000	-20:20
	German2005	0.8	20	2000	-40:40
	German2009	0.67	20	500	-40:40
Semi-Synthesized Data	authorship	0.918	30	100000	-200:200
	bodyfat	0.559	100	100000	-280:280
	calhousing	0.34	20	50000	-40:40
	cpu-small	0.46	50	50000	-60:60
	elevators	0.73	20	50000	-40:40
	fried	0.89	100	100	-2:2
	glass	0.91	100	10000	-240:240
	housing	0.712	25	50000	-20:20
	iris	0.917	100	50000	-1:1
	pendigits	0.86	100	50000	-10:10
	segment	0.916	20	50000	-20:20
	stock	0.834	50	50000	-60:60
	vehicle	0.754	100	10000	-100:100
	vowel	0.85	22	10000	-50:50
	wine	0.90	100	50000	-60:60
	wisconsin	0.61	100	10000	-640:640

[21] Z. Ji, B. Cui, H. Li, Y.-G. Jiang, T. Xiang, T. Hospedales, and Y. Fu, “Deep ranking for image zero-shot multi-label classification,” *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, May 14 2020, date created - 2020-05-15; Date revised - 2020-12-20; Last updated - 2021-01-29. [Online]. Available: http://ezproxy.lib.uts.edu.au/login?url=https://www-proquest-com.ezproxy.lib.uts.edu.au/scholarly-journals/deep-ranking-image-zero-shot-multi-label/docview/2403037643/se-2?accountid=17095

[22] C. Moraga and R. Heider, “"New lamps for old!" (generalized multiple-valued neurons),” in *Proceedings 1999 29th IEEE International Symposium on Multiple-Valued Logic (Cat. No. 99CB36329)*. IEEE, 1999, pp. 36–41.

[23] A. Elgharabawy, “Preference neural network convergence performance,” Video file, 2020. [Online]. Available: https://drive.google.com/drive/folders/1yxuqYoQ3Kiuch-2sLeVe2ocMj12QVsRM?usp=sharing

[24] G. Bologna, “Rule extraction from a multilayer perceptron with staircase activation functions,” 2000.

[25] M. Kendall, “Rank correlation methods,” 1948.

[26] C. Spearman, “The proof and measurement of association between two things,” *The American Journal of Psychology*, vol. 15, no. 1, pp. 72–101, 1904. [Online]. Available: http://www.jstor.org/stable/1412159

[27] W. Cheng, J. Hühn, and E. Hüllermeier, “Decision tree and instance-based learning for label ranking,” in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML ’09. ACM, 2009, p. 161–168.

[28] Y. LeCun and C. Cortes, “MNIST handwritten digit database,” 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[29] A. Krizhevsky, “Learning multiple layers of features from tiny images,” Tech. Rep., 2009.

[30] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms,” 2017, cite arxiv:1708.07747Comment: Dataset is freely available at https://github.com/zalandoresearch/fashion-mnist Benchmark is available at http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/. [Online]. Available: http://arxiv.org/abs/1708.07747

[31] C. R. de Sá and W. Duivesteijn, “Discovering a taste for the unusual: exceptional models for preference mining,” pp. 1775–1807, 2018.

[32] R. Cláudio. (2018) algae dataset. [Online]. Available: http://dx.doi.org/10.17632/3mv94c8jpc.2

[33] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[34] S. Zagoruyko and N. Komodakis, “Wide residual networks,” *ArXiv*, vol. abs/1605.07146, 2016.

[35] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.

[36] M. Grbovic, N. Djuric, S. Guo, and S. Vucetic, “Supervised clustering of label ranking data using label preference information,” *Machine Learning*, vol. 93, no. 2-3, pp. 191–225, 2013.

[37] C. R. de Sá, C. Soares, A. Knobbe, and P. Cortez, “Label ranking forests,” *Expert Syst. J. Knowl. Eng.*, vol. 34, 2017.

[38] Wolfram Research, Inc., Mathematica, “Wolfram.” [Online]. Available: https://www.wolfram.com/mathematica/