Multipartite Pooling
for Deep Convolutional Neural Networks

Arash shahriari
Research School of Engineering
Australian National University
Canberra, Australia
arash.shahriari@anu.edu.au

Fatih Porikli
Research School of Engineering
Australian National University
Canberra, Australia
fatih.porikli@anu.edu.au

Abstract

We propose a novel pooling strategy that learns how to adaptively rank deep convolutional features for selecting more informative representations. To this end, we exploit discriminative analysis to project the features onto a space spanned by the number of classes in the dataset under study. This maps the notion of labels in the feature space into instances in the projected space. We employ these projected distances as a measure to rank the existing features with respect to their specific discriminant power for each individual class. We then apply multipartite ranking to score the separability of the instances and aggregate one-versus-all scores to compute an overall distinction score for each feature. For the pooling, we pick features with the highest scores in a pooling window instead of maximum, average or stochastic random assignments. Our experiments on various benchmarks confirm that the proposed strategy of multipartite pooling is highly beneficial to consistently improve the performance of deep convolutional networks via better generalization of the trained models for the test-time data.

1 Introduction

The considerable complexity of object recognition makes it an interesting research topic in computer vision. Deep neural networks have recently addressed this challenge, with close precision to human observers. They recognize thousands of objects from millions of images, by using the models with large learning capacity. This paper proposes a novel pooling strategy that learns how to rank convolutional features adaptively, allowing the selection of more informative representations.

To this end, the Fisher discrimination is exploited, to project the features into a space, spanned by the number of classes in the dataset under study. This mapping is employed as a measure to rank the existing features, with respect to their specific discriminant power, for each classes. Then, multipartite ranking is applied to score the separability of instances, and to aggregate one-versus-all scores, giving an overall distinction score for each features. For the pooling, features with the highest scores are picked in a pooling window, instead of maximum, average or stochastic random assignments.

Spatial pooling of convolutional features, is critical in many deep neural networks. Pooling aims to select and aggregate features over a local reception field, into a local bag of representations, that are compact and resilient to transformations and distortions of the input [4]. Common pooling strategies often take sum [12], average [21], or maximum [19] response. There are also variants that enhance maximum pooling performance, such as, generalized maximum pooling [24] or fractal maximum pooling [14]. Deterministic pooling can be extended to stochastic alternatives, e.g. random selection of an activation within the pooling region, according to a multinomial distribution [37].
2 Method

There exists a vast literature on instance selection and feature ranking. Instance selection regimes usually belong to either condensation or edition proposals [23]. They attempt to find a subset of data, in which, a trained classifier is provided with, similar or close validation error, as the primary data. Condensed Nearest Neighbour [16], searches for a consistent subset, where every instance inside is assumed to be correctly classified. Some variants of this method are Reduced Nearest Neighbour [13], Selective Nearest Neighbour [29], Minimal Consistent Set [9], Fast Nearest Neighbour Condensation [1], and Prototype Selection by Clustering [27].

In contrast, Edited Nearest Neighbour [36], discards the instances that disagree with the classification responses of their neighbouring instances. Some revisions of this strategy, are Repeated Edited Nearest Neighbour [31], Nearest Centroid Neighbour Edition [30], Edited Normalized Radial Basis Function [18], and Edited Nearest Neighbour Estimating Class Probabilistic and Threshold [33].

On the other hand, the family of feature ranking algorithms can be mainly grouped into, preference learning, multipartite, multilabel, or multilabel ranking. In situations where the instances have only binary labels, the ranking is called bipartite. Different aspects of bipartite ranking have been investigated in numerous studies including, RankBoost [11], RankNet [6], and AUC maximizing [5], which are the ranking versions for AdaBoost, logistic regression, and Support Vector Machines.

There are also several ranking measures, such as, average precision and Normalized Discounted Cumulative Gain. For multilabel instances, multipartite ranking approaches seek to maximize the volume under the ROC surface [35], which is in contrast with the minimization of the pairwise ranking cost [32].

The problem of employing either instance selection or feature ranking methods for pooling in deep neural networks, appears at the testing phase of trained models. The existing ranking algorithms, mostly deals with the training-time ranking. As a result, they are not usually advantageous for the pooling of convolutional features in the test phase. Without pooling, the performance of deep learning architectures degrades substantially. The local feature responses propagate less effectively to neighboring receptive fields, thus the local-global representation power of the convolutional network diminishes. Moreover, the network becomes very sensitive to input deformations.

To tackle the above issues, a novel strategy i.e. multipartite pooling, is introduced in this paper. This ranks convolutional features by employing supervised learning. In supervised learning, the trained scoring function reflects the ordinal relation among class labels. The multipartite pooling scheme learns a projection from the training set. Intuitively, this is a feature selection operator, whose aim is to pick the most informative convolutional features, by learning a multipartite ranking scheme from the training set. Inspired by stochastic pooling, higher ranked activations in each window, are picked with respect to their scoring function responses. Since this multipartite ranking is based on the class information, it can generate a coherent ranking of features, for both of the training and test sets. This also leads to an efficient spread of responses, and effective generalization for deep convolutional neural networks.

In summary, the proposed multipartite pooling method has several advantages. It considers the distribution of each class and calculates the rank of individual features. Due to the data-driven process of scoring, the performance gap between training-test errors, is considerably closer. It generates superior performance on standard benchmark datasets, in comparison with the average, maximum and stochastic pooling schemes, when identical evaluation protocols are applied. The conducted experiments on various benchmarks, confirm that the proposed strategy of multipartite pooling, consistently improves the performance of deep convolutional networks, by using better model generalization for the test-time data.
Figure 1: Feature vs instance ranking. A set of features (columns) and instances (rows) are assigned to $|\mathcal{L}|$ different labels. They are ranked upon their separability, represented by different line patterns and are scored. These scores are used for selecting the best features or instances. To employ either of them for convolutional pooling, the labels must be known. The problem is that, classic feature-instance ranking methods are specific to the training-time data, and there is no way to exploit them for pooling of the test-time data. To solve this inconsistency, the notion of labels is mapped to the test data and then, instance ranking strategies are applied to the pooling layers. This is accomplished by the supervised projection.

3 Formulation

This section begins with multipartite ranking and moves towards the multipartite pooling. The multipartite ranking means scoring of each representation in the feature set, with respect to the distinctive information. Instances with higher scores are expected to be more informative, and hence, receive higher ranks. The intuition of multipartite pooling is about picking the activation instances with the higher scores (ranks) in a pooling window, to achieve better activations in the pooling layer. A graphical interpretation of feature vs instance ranking is depicted in Figure 1, where columns represent the activations.

For a two-class regime, the criterion to calculate the significance of each feature, can be selected from statistical measures, such as, absolute value two-sample t-test with pooled variance estimate $[17]$; relative entropy or Kullback-Leibler distance $[15]$; minimum attainable classification error or Chernoff bound $[26]$; area between the empirical Receiver Operating Characteristic (ROC) curve and the random classifier slope $[10]$; and absolute value of the standardized u-statistic of a two-sample unpaired Wilcoxon test or Mann-Whitney test $[3]$.

Suppose that a set of instances $\mathcal{X} = \{X_1, \ldots, X_{|X|}\}$ are assigned to a set of predefined label $\mathcal{L} = \{L_1, \ldots, L_{|\mathcal{L}|}\}$, such that, $\mathcal{X}$ is a matrix with $|\mathcal{X}|$ instances (rows). The aim is to rank the features (columns), using an independent evaluation criterion. This criterion is a distance that measures the significance of an instance, for imposing higher class distinction in the set $\mathcal{X}$. The absolute value of the criterion for a bipartite ranking scenario, with only two valid labels $\{L_1, L_2\}$, is defined as,

$$C_B(\mathcal{X}) = KL_{12}(X_1, \ldots, X_{|\mathcal{X}|})$$  \hspace{1cm} (1)
The projected set $\mathcal{P}$ is ranked and the scores are aggregated to compute overall criterion $C_M(\mathcal{P})$. Since the columns represent classes inside the feature set $\mathcal{X}$, bipartite rank of each columns is calculated with respect to the rest of columns. This generates $c = |\mathcal{L}|$ different scores, represented by different line patterns, for each of the $|\mathcal{P}|$ instances. By sliding an accumulative bar, represented by grey rectangle, the overall score is computed for each instances. These overall scores are used to rank and pool the most informative instances, which are activations of the pooling layers.

Here, $KL$ is the Kullback-Leibler divergence and $C_B(\mathcal{X})$ is the binary criterion measured for each feature (column) of the set $\mathcal{X}$. This equation can be extended to the summation of binary criteria, where each labels is considered as primary label (foreground) and the rest are merged as secondary labels (background).

The overall criterion of the multipartite case, with multiple labels $\mathcal{L}$, can be formulated as,

$$C_M(\mathcal{X}) = \sum_{i=1, j \neq i}^{i=|\mathcal{L}|} KL_{ij}(X_1, \ldots, X_{|\mathcal{X}|})$$

(2)

where $KL_{ij}$ is the cumulative Kullback-Leibler distance of label $L_i$ to the rest of the labels of set $\mathcal{L}$, which are $\forall L_j \in \mathcal{L} - L_i$. It is clear that, higher values of $C_M(\mathcal{X})$ for a feature, means better class separability. A high-ranked representation is beneficial to any classifier, because there are better distinctions between classes in the set $\mathcal{X}$.

It is possible to employ the above formulation for instance ranking. In other words, instances (rows) are ranked instead of features (columns), which is required for pooling operation, where high-ranked instances are selected as the representations for each convolutional filters. In contrast with feature ranking, the rows of set $\mathcal{X}$, which correspond to convolutional representations, are ranked in the pooling layers.

To connect the features into instances, a projection from the feature space into a new instance space, spanned by the number of classes in $\mathcal{X}$, is employed. In this space, a new set $\mathcal{P}$ is created by multiplying the feature set $\mathcal{X}$ with a projection matrix $A$, such that,

$$C_M(\mathcal{P}) = \sum_{i=1, j \neq i}^{i=|\mathcal{L}|} KL_{ij}(X_1A, \ldots, X_{|\mathcal{X}|A})$$

(3)

where the set $\mathcal{P}$ is a matrix with $|\mathcal{X}|$ instance (rows) and $|\mathcal{L}|$ features (columns). The projection matrix $A$ enables the same ranking strategies for features of the set $\mathcal{X}$, to be applied to the instances of the $\mathcal{P}$, so that, the highly-ranked activations are selected.

Figure 2: Multipartite ranking. The projected set $\mathcal{P}$ is ranked and the scores are aggregated to compute overall criterion $C_M(\mathcal{P})$. Since the columns represent classes inside the feature set $\mathcal{X}$, bipartite rank of each columns is calculated with respect to the rest of columns. This generates $c = |\mathcal{L}|$ different scores, represented by different line patterns, for each of the $|\mathcal{P}|$ instances. By sliding an accumulative bar, represented by grey rectangle, the overall score is computed for each instances. These overall scores are used to rank and pool the most informative instances, which are activations of the pooling layers.
Algorithm 1 Supervised Projection

**Input:** feature set $X$
**Output:** projection matrix $A$

1: Compute $S_w$ (Equation 5) and $S_b$ (Equation 6)
2: Set $A^{(0)}$ as largest eigenvalues of $S_w^{-1}S_b$
3: Minimize Equation 7 by using Equation 10

### 3.1 Supervised Projection

To formulate the above projection, the matrix $A$ can be considered as a mapping, which tries to project $X$ into a space with $c = |L|$ dimensions. The projection matrix $A$ is determined to minimize the Fisher criterion given by,

$$J(A) = tr\left[(AS_wA^T)(AS_bA^T)^{-1}\right]$$  \hspace{1cm} (4)

that $tr(.)$ is the diagonal summation operator. The within ($S_w$) and between ($S_b$) class scatterings are defined as,

$$S_w = \sum_{j=1}^{c} \sum_{x_i \in C_j} (x_i - \mu_j)(x_i - \mu_j)^T$$ \hspace{1cm} (5)

$$S_b = \sum_{j=1}^{c} (\mu_j - \bar{\mu})(\mu_j - \bar{\mu})^T$$ \hspace{1cm} (6)

where $c$, $\mu_j$ and $\bar{\mu}$ are number of classes, mean over class $C_j$ and mean over the set $X$. The matrix $S_w$ can be regarded as average class-specific covariance, whereas $S_b$ can be viewed as the mean distance between all different classes. The purpose of Equation 4 is to maximize the between-class scattering, while preserving within-class dispersion. The solution can be retrieved from a generalized eigenvalue problem $S_bA = \lambda S_wA$. For $c = |L|$ classes, the projection matrix $A$ builds upon eigenvectors corresponding to the largest $c$ eigenvalues of $S_w^{-1}S_b$.[2]

To yield better distinction, the ratio of between and within class scatterings (quotient-of-trace) is minimized [8], by imposing orthogonality to the following cost function,

$$Q(A) = tr(AS_wA^T)\left[tr(AS_bA^T)\right]^{-1} + \lambda||I - AA^T||_2$$ \hspace{1cm} (7)

The first part of function $Q(A)$ defines a form of Fisher criterion that aims for making the highest possible separability among classes. The second term is a regularization term imposing orthogonality into the projection matrix $A$. Looking back at Equation 4 it can be seen that the set of eigenvectors corresponding to the largest $c$ eigenvalues of $S_w^{-1}S_b$ is a solution for the above optimization problem. This can be taken as an initial projection matrix $A^{(0)}$.

Now, it is possible to minimize $Q(A)$ by using stochastic gradient descent. Here, $A^{(0)}$ is employed as an initialization point, because conventional Fisher criterion is the trace-of-quotient, which can be solved by generalized eigenvalue method. Equation 7 is the quotient-of-trace, that requires a different solution [8].
Algorithm 2 Multipartite Ranking

**Input:** feature set \( \mathcal{X} \), label set \( \mathcal{L} \)

**Output:** overall criterion \( C_M(P) \)

1: Compute the projection matrix \( A \) (Algorithm 1)
2: Calculate the projected set \( P = \mathcal{X}A \)

\[
\text{for } i = 1 \text{ to } |\mathcal{L}| \text{ do} \\
3: \text{Split } P \text{ between labels } \{ L_i, L_j \}, \text{ when } L_j \in \mathcal{L} - L_i \\
4: \text{Calculate } KL_{ij}(P) \text{ (Equation 3)} \\
\text{end for} \\
5: \text{Set } C_M(P) = \sum KL_{ij}(P)
\]

To work out the closed form derivatives of Equation 7, suppose that \( Q(A) \) is composed of \( Q_1(A) \) and \( Q_2(A) \) as follows,

\[
Q_1(A) = tr(AS_wA^T)[tr(AS_bA^T)]^{-1} \\
Q_2(A) = \lambda \|I - AA^T\|_2
\]  

According to matrix calculus [28],

\[
\frac{\partial Q}{\partial A} = \frac{tr(AS_wA^T)}{[tr(AS_bA^T)]^2}(S_b^T + S_b)A^T \\
- \frac{1}{tr(AS_bA^T)}(S_w^T + S_w)A^T \\
- \frac{2\lambda}{\|I - AA^T\|_2}A^T(I - AA^T)
\]  

The computation of \( A \) is summarized in Algorithm 1. For implementation purposes, the built-in function of Matlab optimization toolbox [7] is employed.

### 3.2 Multipartite Ranking

Drawing upon the above information, it is possible to put forward the proposed multipartite ranking scheme. Using the instance ranking strategy, one can take the feature set \( \mathcal{X} \), deploy the supervised projection \( A \) (Algorithm 1) to produce the projected set \( P \), and calculate the cumulative Kullback-Leibler distance (Equation 3), as the ranking scores, for each instance of the projected set \( P \).

Since the number of instances in \( P \) is equal to the number of instances in \( \mathcal{X} \), and these two matrices are related linearly through \( A \) via \( P = \mathcal{X}A \), the overall criterion \( C_M(P) \) is sorted to rank the instances of the set \( \mathcal{X} \), in regard to their class separability. Algorithm 2 represents the multipartite ranking method. The process of the multipartite ranking is also visualized in Figure 2. Each column of the set \( P \) represents a specific class of the set \( \mathcal{X} \) and hence, the Kullback-Leibler binary scoring scheme (one-versus-all) is employed to set a criterion measure for each of its individual instances (rows). After it has been applied to all the columns, it starts to scan rows and accumulate scores, resulting in the overall criteria, \( C_M(P) \). This is then used to rank the instances of the projected set \( P \).

When \(\mathbf{X}\) as a numerical example, consider MNIST dataset with 10 classes. The first pooling layer is fed by a stack \(\mathbf{S}\) consisting of convolution responses of \(d = 20\) filters, with \(n = 100\) frames, of size \(w = 24\) by \(h = 24\) pixels. First, \(\mathbf{S}\) is reshaped to 100 samples of size \(576 \times 20\) pixels, form the set \(\mathcal{X}\), which will be concatenated as a \(57600 \times 20\) array. Second, the projection matrix \(\mathbf{A} \in \mathbb{R}^{20 \times 10}\) is computed to project \(\mathcal{X}\). Third, \(C_M(\mathcal{P}) \in \mathbb{R}^{57600 \times 1}\) is computed and partitioned into 100 criterion measures \(C_{M_i}\), of size \(24 \times 24\) pixels. For the pooling of \(\mathcal{S}\), a \(2 \times 2\) window moves along each frame and picks the top-score pixels. The output is a set of 100 features of size \(12 \times 12\) pixels for each of \(20\) convolutional filters.

The reason for projecting to the space spanned by the number of classes, is to use the above, one-versus-all strategy. The bipartite ranking by Kullback-Leibler divergence requires that, one main class is selected as the foreground label, while those remaining are used as background labels. It gives a measure of how the foreground is separated from the background data. It is necessary to use statistics to ensure that the cumulative criterion, \(C_M(\mathcal{P})\) is a true representation of the all instances, contained within the set \(\mathcal{X}\).

When \(\mathcal{X}\) is projected to lower dimensions than the number of available classes, the result is that, some of the classes are missed. In contrast, projection of \(\mathcal{X}\) to higher dimensions than the number of classes, leads to partitioning of some classes to pseudo labels, that are not queried during the test phase. Either way, the generated scores are not reliable for the sake of pooling, because they are not derived from the actual distribution of the classes.

### 3.3 Multipartite Pooling

The above multipartite ranking strategy can be employed for the pooling. In general, a deep convolutional neural network consists of consecutive convolution and pooling layers. The convolutional layers extract common patterns within local patches. Then, a nonlinear elementwise operator is applied to the convolutional features, and the resulting activations are passed to the pooling layers. These activations are less sensitive to the precise locations of structures within the data, than the primary features. Therefore, the consecutive convolutional layers can extract features, that are not susceptible to spatial transformations or distortions of the input \([37]\).

Suppose that a stack of convolutional features \(\mathbf{S} = \{S_1, \ldots, S_{|\mathcal{S}|}\}\) passes through the pooling layer. The matrix \(\mathcal{S}\) is an array of dimensions \(h \times w \times d \times n\) where \(h\) and \(w\) are height and width of samples, \(d\) is depth of stack (number of filters), such that \(S_i \in \mathbb{R}^{h \times w \times d}\) creates a three-dimensional vector, and \(n\) is number of samples in the stack \((n = |\mathcal{S}|)\).

The standard pooling methods either retain the maximum or average value, over the pooling region per channel. The multipartite pooling method begins with the reshaping of feature stack \(\mathcal{S}\) to form \(n\) two-dimensional \(X_i \in \mathbb{R}^{h \times w}\). The elements of set \(\mathcal{X} = \{X_1, \ldots, X_{|\mathcal{X}|}\}\) are concatenated such that \(|\mathcal{X}| = n\) and \(\mathcal{X} \in \mathbb{R}^{h \times w \times d}\) is a two-dimensional matrix. Now, \(\mathcal{X}\) is ready to deploy Algorithm 2 and compute the overall criterion \(C_M(\mathcal{P}) \in \mathbb{R}^{h \times w \times 1}\). Partitioning to \(n\) and reshaping into \(h \times w\) windows, give \(C_M(\mathcal{S}) = \{C_{M_1}, \ldots, C_{M_{|\mathcal{P}|}}\}\), that \(C_{M_i} \in \mathbb{R}^{h \times w}\) provides the rank of each pixels of \(X_i\). To apply the pooling, a sliding window goes through each region and picks the representation with the greatest \(C_{M_i}(\mathcal{S})\). These are the activations with the best separation among available classes.

As a numerical example, consider MNIST dataset with \(|\mathcal{L}| = 10\) classes. The first pooling layer is fed by a stack \(\mathcal{S}\), consisting of convolution responses of \(d = 20\) filters, with \(n = 100\) frames, of size \(w = 24\) by \(h = 24\) pixels. First, \(\mathcal{S}\) is reshaped to 100 samples of size \(576 \times 20\) pixels, form the set \(\mathcal{X}\), which will be concatenated as a \(57600 \times 20\) array. Second, the projection matrix \(\mathbf{A} \in \mathbb{R}^{20 \times 10}\) is calculated to project \(\mathcal{X}\). Third, \(C_M(\mathcal{P}) \in \mathbb{R}^{57600 \times 1}\) is computed and partitioned into 100 criterion measures \(C_{M_i}\), of size \(24 \times 24\) pixels. For the pooling of \(\mathcal{S}\), a \(2 \times 2\) window moves along each frame and picks the top-score pixels. The output is a set of 100 features of size \(12 \times 12\) pixels for each of \(20\) convolutional filters.

---

**Algorithm 3 Multipartite Pooling**

**Input:** convolutional feature stack \(\mathcal{S} \in \mathbb{R}^{h \times w \times d \times n}\)

**Output:** overall criterion \(C_M(\mathcal{S})\)

1. Reshape each \(S_i \in \mathbb{R}^{h \times w \times d}\) to \(S_i \in \mathbb{R}^{h \times w \times d}\)
2. Concatenate all \(S_i\) columns to give \(\mathcal{X} \in \mathbb{R}^{h \times w \times d}\)
3. Calculate \(C_M(\mathcal{P}) \in \mathbb{R}^{h \times w \times 1}\) by Algorithm 2
4. Partition \(C_M(\mathcal{P})\) to give \(C_M(\mathcal{S}) \in \mathbb{R}^{h \times w \times n}\)
5. Pool the activations based on \(C_{M_i} \in \mathbb{R}^{h \times w}\) for all \(i \in [1 : n]\)
4 Experiments

For evaluation purposes, the multipartite pooling is compared with the popular maximum, average and stochastic poolings. A standard experimental setup [37] is followed to apply the multipartite pooling for MNIST, CIFAR and Street View House Numbers (SVHN) datasets. The results show that, when multipartite pooling is employed to pool convolutional features, lower test error rates than other pooling strategies, are achieved. For implementation, the library provided by the Oxford Visual Geometry Group [34] is used.

4.1 Datasets

The MNIST dataset [22], contains 60,000 training examples, and 10,000 test samples, normalized to $20 \times 20$ pixels, centred by centre of mass in $28 \times 28$ pixels, and sheared by horizontally shifting, such that, the principal axis is vertical. The foreground pixels were set to one, and the background to zero.

The CIFAR dataset [20], includes two subsets. The first subset, CIFAR-10 consists of 10 classes of objects with 6,000 images per class. The classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. It was divided into 5,000 randomly selected images per class as training set, and the rest served as test samples. The second subset, CIFAR-100 has 600 images for each of 100 classes. These classes also come in 20 super-classes, each consisting of five classes.

The SVHN dataset [25], was extracted from a large number of Google Street View images by automated algorithms and the Amazon Mechanical Turk (AMT). It consists of over 600,000 labelled characters in full numbers and MNIST-like cropped digits in $32 \times 32$ pixels. Three subsets are available containing 73,257 digits for training, 26,032 for testing and 531,131 extra samples.

4.2 Results & Discussion

The following tables represent the evaluation results for four different schemes, including max, average, stochastic, and multipartite poolings, in terms of the training and test errors. To gain an insight into added computational load, one should recall that the only intensive calculations are related to working out the supervised projection, at the training phase. This procedure (Algorithm 1), depends on the number of activations ($h \times w \times n$ instances) and convolutional filters ($d$ dimensions), which are employed in Algorithm 3. For testing, one just multiplies and no computational cost is involved. For example, the number of samples, trained in an identical architecture for MNIST dataset, are 285, and 195 samples per second for the max and multipartite pooling strategies, respectively.

The classification performance on MNIST dataset is reported in Table 1. It can be seen that, max pooling gives the best training performance, but its test error is larger than the stochastic and multipartite pooling. In other words, it may overfit for MNIST, although its test performance is higher than the average pooling. The multipartite pooling performs better than all other schemes, despite greater training error, compared to max and stochastic pooling.
Table 1: Classification errors for different pooling strategies on MNIST dataset. The multipartite pooling approach provides the lowest test error, in spite of high training error. This is due to better generalization of the proposed pooling, compared to the other methods.

| Pooling           | Train (%) | Test (%) |
|-------------------|-----------|----------|
| Average           | 0.57      | 0.83     |
| Max               | **0.04**  | 0.55     |
| Stochastic        | 0.33      | 0.47     |
| Multipartite (Proposed) | 0.38 | **0.41** |

Table 2: Classification errors for different pooling strategies on CIFAR-10 and CIFAR-100 datasets. The multipartite pooling outperforms other pooling schemes on test errors, but it is behind on training errors. The close gap between training and test errors, leads to better classification performance for the proposed pooling strategy.

| Pooling           | Train (%) | Test (%) |
|-------------------|-----------|----------|
| Average           | 1.92      | 19.24    |
| Max               | **0.0**   | 19.40    |
| Stochastic        | 3.40      | 15.13    |
| Multipartite (Proposed) | 12.63 | **13.45** |

| Pooling           | Train (%) | Test (%) |
|-------------------|-----------|----------|
| Average           | 11.20     | 47.77    |
| Max               | **0.17**  | 50.90    |
| Stochastic        | 21.22     | 42.51    |
| Multipartite (Proposed) | 36.32 | **40.81** |

Table 3: Classification errors for different pooling strategies on SVHN dataset. The multipartite pooling scheme gives the best performance on the test, that is closely followed by the training error.

| Pooling           | Train (%) | Test (%) |
|-------------------|-----------|----------|
| Average           | 1.65      | 3.72     |
| Max               | **0.13**  | 3.81     |
| Stochastic        | 1.41      | 2.80     |
| Multipartite (Proposed) | 2.09 | **2.15** |

The multipartite pooling is more competent because it draws upon the statistics of training-test data for pooling. It is in contrast to picking the maximum response (max pooling); smoothing the activations (average pooling); and random selection (stochastic pooling); which disregard the data distribution. In a pooling layer of any deep learning architecture, aggregation of the best available features is critical to infer complicated contexts, such as, shape derived from primitives of edge and corners. The proposed pooling learns how to pick the best informative activations from the training set, and then, employs this knowledge at the test phase. As a result, the performance consistently improves in all the experiments.

Figures 4 and 5 show the training-test performances of MNIST dataset for 20 epochs. Except for the early epochs, they are quite close to each other in the multipartite pooling regime. The reason is that, both of the training-test poolings are connected with a common factor; the projection matrix $A$. This is trained with the training set and is deployed by multipartite pooling on the test set, to pick the most informative activations. Since the same criterion (Kullback-Leibler) is employed to rank the projected activations for training-test, and they are mapped with the same projection matrix $A$, it is expected that the trained network will demonstrate better generalization than alternative pooling schemes, where the training-test poolings are statistically disconnected. The graphs show that the multipartite pooling generalizes considerably better.
Tables 2 provides the performance metrics for CIFAR datasets. It is apparent that the multipartite pooling outperforms other approaches on the test performance. It also prevents the model from overfitting, in contrast to the max pooling. In the proposed pooling method, better generalization also contributes to another advantage; preventing under-overfitting. As mentioned before, pooling at the test phase is linked to the training phase by the projection. This ensures that the test performance follows the training closely and hence, it is less likely to end up under-overfitting.

One striking observation is that the gap between the training-test errors is wider for CIFAR-100 compared to CIFAR-10. This relates to the number of classes in each subset of CIFAR. Since CIFAR-100 has more classes, it is more difficult to impose separability, hence, the difference between the training and test performances will increase. Figures 6 and 7 depicts the errors of CIFAR-10 dataset for 50 epochs. It is clear that employing of the max pooling results in a huge gap between the training-test performances due to the overfitting.

Finally, the evaluation outcomes for SVHN dataset are presented in Table 3. Again, the multipartite pooling does a better job on the test. The error is close to the stochastic pooling method. This implies that when the number of samples in a dataset increases greatly, the multipartite ranking scores lean towards the probability distribution, generated by the stochastic pooling. Here, any infinitesimal numerical errors may also lead to an inaccurate pooling, which may, in return, degrade the informative activations of a layer for both of the pooling methods. Since the multipartite pooling takes a deterministic approach, the effect of numerical inconsistencies, is considerably smaller than stochastic pooling, which randomly picks activations on a multinomial distribution [37].

Overall, the employment of multipartite ranking for the purpose of pooling in deep convolutional neural networks, produces superior results compared to all the other strategies, tested in the experiments. It is robust to overfitting and shows better generalization characteristics, by connecting the training and test statistics.

5 Conclusion

We introduce a novel pooling strategy called multipartite pooling that is based on multipartite ranking of the features in pooling layers of deep convolutional neural networks. This pooling scheme projects the features to a new space and then, score them by an accumulative bipartite ranking approach. These scores can be used to pick the highly informative activations in the pooling layers of any deep convolutional neural networks. We conduct our experiments on four publicly available datasets (MNIST, CIFAR-10, CIFAR-100, and SVHN) and report the errors of four different pooling schemes (maximum, average, stochastic, and multipartite). The results show that our proposed multipartite pooling method outperforms all other pooling strategies in all datasets and provides a more efficient generalization for the deep learning architectures.
Figure 4: Max pooling for MNIST. The first graph represents the loss function (objective) for both training (train) and test (val) on MNIST dataset. The other graphs correspond to the top 1 (top1err) and the top 5 (top5err) errors.

Figure 5: Multipartite pooling for MNIST. In comparison with the max pooling (Figure 4), the test loss and errors (val) are reduced by applying the multipartite pooling technique. Note that the training and test performances get closer to each other, indicating better generalization of the trained network.
Figure 6: Max pooling for CIFAR-10. Compared to MNIST, this results in greater loss and errors on CIFAR-10, due to the variety in samples and tasks (character vs object recognition). The gap between the training (train) and test (val) errors, is considerably wider for CIFAR-10.

Figure 7: Multipartite pooling for CIFAR-10. Besides smaller losses and errors, with respect to the max pooling (Figure 6), the training (train) and test (val) performances indicate closer gaps, due to better generalization of the model, using by multipartite pooling.
References

[1] F. Angiulli. Fast nearest neighbor condensation for large data sets classification. *IEEE Transactions on Knowledge and Data Engineering*, 19(11):1450–1464, 2007.

[2] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[3] L. L. Bohn and D. A. Wolfe. The effect of imperfect judgment rankings on properties of procedures based on the ranked-set samples analog of the mann-whitney-wilcoxon statistic. *Journal of the American Statistical Association*, 89(425):168–176, 1994.

[4] Y.-L. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118, 2010.

[5] U. Brefeld and T. Scheffer. Auc maximizing support vector learning. In *Proceedings of the ICML 2005 workshop on ROC Analysis in Machine Learning*. Citeseer, 2005.

[6] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005.

[7] T. F. Coleman and Y. Li. A reflective newton method for minimizing a quadratic function subject to bounds on some of the variables. *SIAM Journal on Optimization*, 6(4):1040–1058, 1996.

[8] J. P. Cunningham and Z. Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *Journal of Machine Learning Research*, 2015.

[9] B. V. Dasarathy. Minimal consistent set (mcs) identification for optimal nearest neighbor decision systems design. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(3):511–517, 1994.

[10] T. Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[11] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of machine learning research*, 4(Nov):933–969, 2003.

[12] K. Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

[13] K. C. Gowda and G. Krishna. The condensed nearest neighbor rule using the concept of mutual nearest neighborhood. *IEEE Transactions on Information Theory*, 25(4):488–490, 1979.

[14] B. Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

[15] J. R. Hershey and P. A. Olsen. Approximating the kullback leibler divergence between gaussian mixture models. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP’07*, volume 4, pages IV–317. IEEE, 2007.

[16] C. Hilborn. Diffusion process in a dynamical system. *IEEE transactions on information theory*, 1968.

[17] N. Jain, J. Thatte, T. Braciale, K. Ley, M. O’Connell, and J. K. Lee. Local-pooled-error test for identifying differentially expressed genes with a small number of replicated microarrays. *Bioinformatics*, 19(15):1945–1951, 2003.

[18] N. Jankowski and M. Grochowski. Comparison of instances selection algorithms i. algorithms survey. In *International conference on artificial intelligence and soft computing*, pages 598–603. Springer, 2004.

[19] K. Jarrett, K. Kavukcuoglu, Y. LeCun, et al. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.

[20] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[21] B. B. Le Cun, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*. Citeseer, 1990.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[23] E. Leyva, A. González, and R. Pérez. Three new instance selection methods based on local sets: A comparative study with several approaches from a bi-objective perspective. *Pattern Recognition*, 48(4):1523–1537, 2015.

[24] N. Murray and F. Perronnin. Generalized max pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2473–2480, 2014.

[25] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[26] M. Nussbaun and A. Szkoła. The chernoff lower bound for symmetric quantum hypothesis testing. *The Annals of Statistics*, pages 1040–1057, 2009.

[27] J. A. Olivera-Lóopez, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad. A new fast prototype selection method based on clustering. *Pattern Analysis and Applications*, 13(2):131–141, 2010.
[28] K. B. Petersen, M. S. Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7:15, 2008.
[29] G. Ritter, H. Woodruff, S. Lowry, and T. Isenhour. An algorithm for a selective nearest neighbor decision rule. *IEEE Transactions on Information Theory*, 21(6):665–669, 1975.
[30] J. S. Sánchez, R. Barandela, A. I. Marqués, R. Alejo, and J. Badenas. Analysis of new techniques to obtain quality training sets. *Pattern Recognition Letters*, 24(7):1015–1022, 2003.
[31] I. Tomek. An experiment with the edited nearest-neighbor rule. *IEEE Transactions on systems, Man, and Cybernetics*, (6):448–452, 1976.
[32] K. Uematsu and Y. Lee. Statistical optimality in multipartite ranking and ordinal regression. *IEEE transactions on pattern analysis and machine intelligence*, 37(5):1080–1094, 2015.
[33] F. Vázquez, J. S. Sánchez, and F. Pla. A stochastic approach to wilson’s editing algorithm. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 35–42. Springer, 2005.
[34] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. [http://www.vlfeat.org/](http://www.vlfeat.org/), 2008.
[35] W. Waegeman and B. De Baets. On the era ranking representability of pairwise bipartite ranking functions. *Artificial Intelligence*, 175(7):1223–1250, 2011.
[36] D. L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, (3):408–421, 1972.
[37] M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*, 2013.