SoDeep: a Sorting Deep net to learn ranking loss surrogates

Martin Engilberge[1,2], Louis Chevallier[2], Patrick Pérez[3], Matthieu Cord[1,3]

[1]Sorbonne Université, Paris, France, [2]Technicolor, Cesson Sévigné, France, [3]Valeo.ai, Paris, France

{martin.engilberge, matthieu.cord}@lip6.fr patrick.perez@valeo.com louis.chevallier@technicolor.com

Abstract

Several tasks in machine learning are evaluated using non-differentiable metrics such as mean average precision or Spearman correlation. However, their non-differentiability prevents from using them as objective functions in a learning framework. Surrogate and relaxation methods exist but tend to be specific to a given metric.

In the present work, we introduce a new method to learn approximations of such non-differentiable objective functions. Our approach is based on a deep architecture that approximates the sorting of arbitrary sets of scores. It is trained virtually for free using synthetic data. This sorting deep (SoDeep) net can then be combined in a plug-and-play manner with existing deep architectures. We demonstrate the interest of our approach in three different tasks that require ranking: Cross-modal text-image retrieval, multi-label image classification and visual memorability ranking. Our approach yields very competitive results on these three tasks, which validates the merit and the flexibility of SoDeep as a proxy for sorting operation in ranking-based losses.

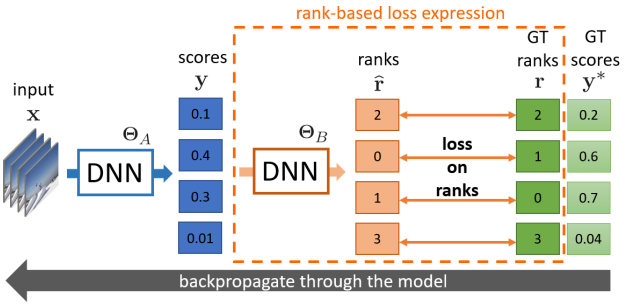


Figure 1: **Overview of SoDeep, the proposed end-to-end trainable deep architecture to approximate non-differentiable ranking metrics**. A pre-trained differentiable sorter (deep neural net [DNN] Θ_B) is used to convert into ranks the raw scores given by the model (DNN Θ_A) being trained to a collection of inputs. A loss is then applied to the predicted rank and the error can be back-propagated through the differentiable sorter and used to update the weights Θ_A.

1. Introduction

Deep learning approaches have gained enormous research interest for many Computer Vision tasks in the recent years. Deep convolutional networks are now commonly used to learn state-of-the-art models for visual recognition, including image classification [26, 18, 35] and visual semantic embedding [25, 22, 37]. One of the strengths of these deep approaches is the ability to train them in an end-to-end manner removing the need for handcrafted features [29]. In such a paradigm, the network starts with the raw inputs, and handles feature extraction (low level and high-level features) and prediction internally. The main requirement is to define a trainable scheme. For deep architectures, stochastic gradient descent with back-propagation is usually performed to minimize an objective function. This loss function depends on the target task but has to be at least differentiable.

Machine learning tasks are often evaluated and compared using metrics which differ from the objective function used during training. The choice of an evaluation metric is intimately related to the definition of the task at hand, even sometimes to the benchmark itself. For example, accuracy seems to be the natural choice to evaluate classification methods, whereas the choice of the objective function is also influenced by the mathematical properties that allow a proper optimization of the model. For classification, one would typically choose the cross entropy loss – a differentiable function – over the non-differentiable accuracy. Ideally, the objective function used during training would be identical to the evaluation metric. However, standard evaluation metrics are often not suitable as training objectives for lack of differentiability to start with. This results in the use of surrogate loss functions that are better behaved (smooth, possibly convex). Unfortunately, coming up with good surrogate functions is not an easy task.

In this paper, we focus on the non-differentiability of the evaluation metrics used in ranking-based tasks such as recall, mean average precision and Spearman correlation. Departing from prior art on building surrogates losses for such tasks, we adopt a simple, yet effective, learning approach: Our main idea is to approximate the non-differentiable part of such ranking-based metrics by an all-purpose learnable deep neural network. In effect, this architecture is designed and trained to mimic sorting operations. We call it SoDeep. SoDeep can be added in a plug-and-play manner on top of any deep network trained for tasks whose final evaluation metric is rank-based, hence not differentiable. The resulting combined architecture is end-to-end learnable with a loss that relates closely to the final metric.

Our contributions are as follows:

- We propose a deep neural net that acts as a differentiable proxy for ranking, allowing one to rewrite different evaluation metrics as functions of this sorter, hence making them differentiable and suitable as training loss.
- We explore two types of architectures for this trainable sorting function: convolutional and recurrent.
- We combine the proposed differentiable sorting module with standard deep CNNs, train them end-to-end on three challenging tasks, and demonstrate the merit of this novel approach through extensive evaluations of the resulting models.

The rest of the paper is organized as follows. We discuss in Section 2 the related works on direct and indirect optimization of ranking-based metrics, and position our work accordingly. Section 3 is dedicated to the presentation of our approach. We show in particular how a "universal" sorting proxy suffices to tackle standard rank-based metrics, and present different architectures to this end. More details on the system and its training are reported in Section 4, along with various experiments. We first establish new state-of-the-art performance on cross-modal retrieval, then we show the benefits of our learned loss function compared to standard methods on memorability prediction and multi-label image classification.

2. Related works

Many data processing systems rely on sorting operations at some stage of their pipeline. It is the case also in machine learning, where handling such non-differentiable, non-local operations can be a real challenge [32]. For example, retrieval systems require to rank a set of database items according to their relevance to a query. For sake of training, simple loss functions that are decomposable over each training sample have been proposed as for instance in [19] for the area under the ROC curve. Recently, some more complex non-decomposable losses (such as the Average Precision (AP), Spearman coefficient, and normalized discounted cumulative gain (nDCG) [3]) that present hard computational challenges have been proposed [31].

Mean average precision optimization Our work shares the high level goal of using ranking metrics as training objective function with many works before us. Several works studied the problem of optimizing average precision with support vector machines [21, 40] and other works extended these approaches to neural networks [1, 31, 8]. To learn rank, the seminal work [21] relies on a structured hinge upper bound to the loss. Further works reduce the computational complexity [31] or rely on asymptotic methods [36]. The focus of these works is mainly on the relaxation of the mean average precision, while our focus is on learning a surrogate for the ranking operation itself such that it can be combined with multiple ranking metrics. In contrast to most ranking-based techniques, which have to face the high computational complexity of the loss augmented inference [21, 36, 31], we propose a fast, generic, deep sorting architecture that can be used in gradient-based training for rank-based tasks.

Application of ranking based metrics Ranking is commonly used in evaluation metrics. On retrieval tasks such as cross-modal retrieval [25, 22, 15, 12, 30], recall is the standard evaluation. Image classification [11, 9] and object recognition are evaluated with mean average precision in the multi-label case. Ordinal regression [5] is evaluated using Spearman correlation.

Existing surrogate functions Multiple surrogates for ranking exist. Using metric learning to do retrieval is one of them. This popular approach avoids the use of the ranking function altogether. Instead, pairwise [39], triplet-wise [38, 4] and list-wise [13, 2] losses are used to optimize distances in a latent space. The cross-entropy loss is typically used for multi-label and multi-class classification tasks.

3. SoDeep approach

Rank-based metrics such as recall, Spearman correlation and mean average precision can be expressed as a function of the rank of the output scores. The computation of the rank being the only non-differentiable part of these metrics, we propose to learn a surrogate network that approximates directly this sorting operation.

3.1. Learning a sorting proxy

Let $\mathbf{y} \in \mathbb{R}^d$ be a vector of d real values and $\mathbf{rk}$ the ranking function so that $\mathbf{rk}(\mathbf{y}) \in \{1 \cdots d\}^d$ is the vector containing the rank for each variable in $\mathbf{y}$, *i.e.* $\mathbf{rk}(\mathbf{y})_i$ is the rank of $\mathbf{y}_i$ among the $\mathbf{y}_j$'s. We want to design a deep architecture $f_{\mathbf{\Theta}_B}$ that is able to mimic this sorting operator. The training procedure of this DNN is summarized in Fig. 2. The aim is to learn its parameters, $\mathbf{\Theta}_B$, so that the output of

the network is as close as possible to the output of the exact sorting.

Before discussing possible architectures, let's consider the training of this network, independent of its future use. We first generate a training set by randomly sampling N input vectors $\mathbf{y}^{(n)}$ and we compute through exact sorting the associated ground-truth rank vectors $\mathbf{r}^{(n)} = \mathbf{rk}(\mathbf{y}^{(n)})$. We then classically learn the DNN $f_{\mathbf{\Theta}_B}$ by minimizing a L_1 loss between the predicted ranking vector $\hat{\mathbf{r}} = f_{\mathbf{\Theta}_B}(\mathbf{y})$ and the ground-truth rank $\mathbf{r}$ over the training set:

$$\min_{\mathbf{\Theta}_B} \sum_{n=1}^{N} \left\| \mathbf{rk}(\mathbf{y}^{(n)}) - f_{\mathbf{\Theta}_B}(\mathbf{y}^{(n)}) \right\|_1 . \quad (1)$$

We explore in the following different network architectures and we explain how the training data is generated.

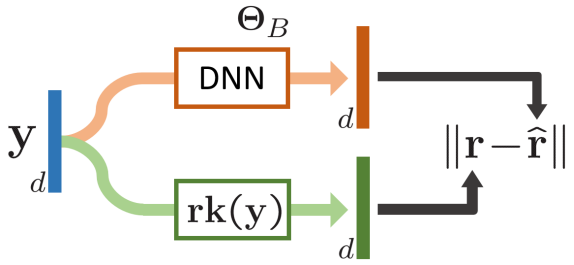


Figure 2: **Training a differentiable sorter**. Given a score vector $\mathbf{y}$ we learn the parameters $\mathbf{\Theta}_B$ of a DNN such that its output $\hat{\mathbf{r}}$ approximates the true rank vector $\mathbf{rk}(\mathbf{y})$. The model is trained using gradient descent and an L_1 loss. Once trained, $f_{\mathbf{\Theta}_B}$ can be used as a differentiable surrogate of the ranking function.

3.1.1 Sorter architectures

We investigate two types of architectures for our differentiable sorter $f_{\mathbf{\Theta}_B}$. One is a recurrent network and the other one a convolutional network, each capturing interesting aspects of standard sorting algorithms:

- The recurrent architecture in Fig. 3a consists of a bi-directional LSTM [34] followed by a linear projection. The bi-directional recurrent network creates a connection between the output of the network and every input, which is critical for ranking computation. Knowledge about the whole sequence is needed to compute the true rank of any element.
- The convolutional architecture in Fig. 3b consists of 8 convolutional blocks, each of these blocks being a one-dimensional convolution followed by a batch normalization layer [20] and a ReLU activation function. The sizes of the convolutional filters are chosen such that the output of the network contains as many channels as the length of the input sequence. Convolutions are used for their local property: indeed, sorting algorithms such as bubble sort [14] only rely on a sequence of local operations. The intuition is that a deep enough convolutional network, with its cascaded local operations, should be able to mimic recursive sorting algorithms and thus to provide an efficient approximation of ranks.

We will further discuss the interest of both types of SoDeep block architectures in the experiments.

3.1.2 Training data

SoDeep module can be easily (pre)trained with supervision on synthetic data. Indeed, while being non-differentiable, the ranking function $\mathbf{rk}$ can be computed with classic sorting algorithms. The training data consists of vectors of randomly generated scalars, associated with their ground-truth rank vectors. In our experiments, the numbers are sampled from different types of distributions:

- Uniform distribution over $[-1, 1]$;
- Normal distribution with $\mu = 0$ and $\sigma = 1$;
- Sequence of evenly spaced numbers in a uniformly drawn random sub-range of $[-1, 1]$;
- Random mixtures of the previous distributions.

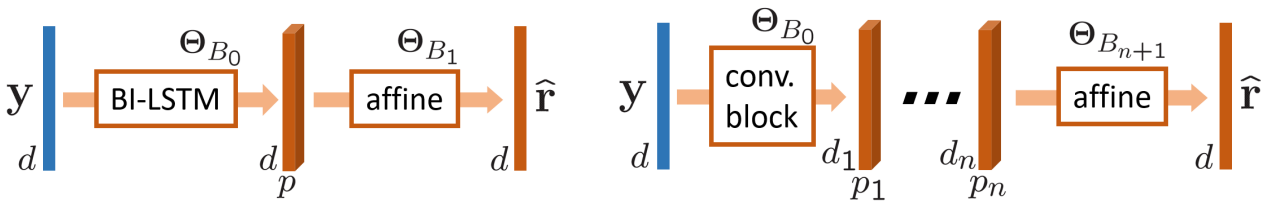
While the differentiable sorter can be trained ahead of time on a variety of input distributions, as explained above, there might be a shift with the actual score distribution that the main network $f_{\mathbf{\Theta}_A}$ will output for the task at hand. This shift can reduce naturally during training, or an alignment can be explicitly enforced. For example, $f_{\mathbf{\Theta}_A}$ can be designed to output data in the interval used to learn the sorter, with the help of bounded functions such as cosine similarity.

3.2. Using SoDeep for training with rank-based loss

Rank-based metrics are used for evaluating and comparing learned models in a number of tasks. Recall is a standard metric for image and information retrieval, mean Average Prediction (mAP) for classification and recognition, and Spearman correlation for ordinal prediction. This type of rank-based metrics are non-differentiable because they require to transition from the continuous domain (score) toward the discrete domain (rank).

As presented in Fig 1, we propose to insert a pre-trained SoDeep proxy block $f_{\mathbf{\Theta}_B}$ between the deep scoring function $f_{\mathbf{\Theta}_A}$ and the chosen rank-based loss. We show in the following how mAP, Spearman correlation and recall can be expressed as functions of the rank and combined with SoDeep accordingly.

In the following we assume a training set of annotated pairs $\{(\mathbf{x}_i, y_i^*)\}_{i=1}^{M}$ for the task at hand. A group $\mathcal{B}$ of



(a) **Architecture of LSTM sorter**.

(b) **Architecture of CNN sorter**.

Figure 3: **SoDeep architecture.** The sorter takes a vector of raw score $\mathbf{y} \in \mathbb{R}^d$ as input and outputs a vector $\hat{\mathbf{r}} \in \mathbb{R}^d$. Two architectures are explored, one recurrent (a), the other one, convolutional (b). Both architectures present a last affine layer to get a final projection to a vector $\hat{\mathbf{r}}$ in $\mathbb{R}^d$. Note that even if it is not explicitly enforced, $\hat{\mathbf{r}}$ will try to mimic as close as possible the vector of the ranks of the $\mathbf{y}$ variables.

d training examples among them yields a prediction vector $\mathbf{y}(\mathbf{\Theta}_A) = [f_{\mathbf{\Theta}_A}(\mathbf{x}_i)]_{i\in\mathcal{B}}$ and an associated ground-truth score vector $\mathbf{y}^* = [y_i^*]_{i\in\mathcal{B}}$ (Fig. 1).

3.2.1 Spearman correlation

For two vectors $\mathbf{y}$ and $\mathbf{y}'$ of size d, corresponding to two sets of d observations, the Spearman correlation [7] is defined as:

$$r_s = 1 - \frac{6\|\mathbf{rk}(\mathbf{y}) - \mathbf{rk}(\mathbf{y}')\|_2^2}{d(d^2-1)}. \tag{2}$$

Maximizing w.r.t. parameters $\mathbf{\Theta}_A$ the sum of Spearman correlations (2) between ground truth and predicted score vectors over N subsets of training examples amounts to solving the minimization problem:

$$\min_{\mathbf{\Theta}_A} \sum_{n=1}^{N} \left\| \mathbf{rk}(\mathbf{y}^{(n)}) - \mathbf{rk}(\mathbf{y}^{*(n)}) \right\|_2^2, \tag{3}$$

with the loss not being differentiable.

Using now our differentiable proxy instead of the rank function, we can define the new Spearman loss for a group $\mathcal{B}$:

$$\mathcal{L}_{SPR}(\mathbf{\Theta}_A, \mathcal{B}) = \sum_{n=1}^{N} \left\| f_{\mathbf{\Theta}_B}(\mathbf{y}(\mathbf{\Theta}_A)^{(n)}) - \mathbf{rk}(\mathbf{y}^{*(n)}) \right\|_2^2. \tag{4}$$

Training will typically minimize it over a large set of groups. Note that here the optimization is done over $\mathbf{\Theta}_A$, knowing that SoDepp block $f_{\mathbf{\Theta}_B}$ has been trained independently on specific synthetic training data. Optionally, the block can be fine-tuned along the way, hence minimizing w.r.t. $\mathbf{\Theta}_B$ as well.

3.2.2 Mean Average Precision (mAP)

Multilabel image classification is often evaluated using mAP, a metric from information retrieval. To define it, each of the C classes is considered as a query over the d elements of the datasets. For class c, denoting $\mathbf{y}_c^*$ the d-dimensional ground-truth binary vector and $\mathbf{y}_c$ the vector of scores for this class, the average precision (AP) for the class is defined as [40] :

$$AP(\mathbf{y}_c, \mathbf{y}_c^*) = \frac{1}{\text{rel}} \sum_{j:\mathbf{y}_c^*(j)=1} Prec(j), \tag{5}$$

where $\text{rel} = |j : \mathbf{y}_c^*(j) = 1|$ is the number of positive items for class c and precision for element j is defined as:

$$Prec(j) = \frac{|\{s \in \mathcal{S} : \mathbf{y}_c^*(s) = 1\}|}{\mathbf{rk}(\mathbf{y}_c)_j}, \tag{6}$$

with $\mathcal{S}$ the set of indices of the elements of $\mathbf{y}_c$ larger than $\mathbf{y}_c(j)$.

Minimizing $\mathbf{rk}(\mathbf{y})_j$ for all j from class c (i.e., those verifying $\mathbf{y}_c^*(j) = 1$) will be used as a surrogate of the maximization of the AP over predictor's parameters $\mathbf{\Theta}_A$.

The mAP is obtained by averaging AP over the C classes. Replacing the rank function by its differentiable proxy, the proposed mAP-based loss reads:

$$\mathcal{L}_{mAP}(\mathbf{\Theta}_A, \mathcal{B}) = \sum_{c=1}^{C} \langle f_{\mathbf{\Theta}_B}(\mathbf{y}_c), \mathbf{y}_c^* \rangle. \tag{7}$$

3.2.3 Recall at K

Recall at rank k is often used to evaluate retrieval tasks. In the following we assume a training set $\{\mathbf{x}_i\}_{i=1}^M$ for the task at hand. A group $\mathcal{B}$ of d training examples among them yields a $d \times d$ prediction matrix $\mathbf{Y}(\mathbf{\Theta}_A) = [f_{\mathbf{\Theta}_A}(\mathbf{x}_i)]_{i\in\mathcal{B}}$ representing the scores of all pairwise combinations of training examples in $\mathcal{B}$. In other words, the i-th column of this matrix, $\mathbf{Y}[i] = f_{\mathbf{\Theta}_A}(\mathbf{x}_i)$, provides the relevance of other vectors in the group w.r.t. to query $\mathbf{x}_i$.

This matrix being given, recall at K is defined as:

$$R@K(\mathbf{Y}) = \frac{1}{d} \sum_{i=1}^{d} \begin{cases} 1, & \text{if } \mathbf{rk}(\mathbf{Y}[i])_p < K \\ 0, & \text{otherwise,} \end{cases} \tag{8}$$

with p the index of the unique positive entry in $\mathbf{Y}[i]$, a single relevant item being assumed for query $\mathbf{x}_i$.

Once again, our sorter enables a differentiable implementation of this measure. However, we could not obtain conclusive results yet, possibly due to the batch size limiting the range of the summation. We found, however, an alternative way to leverage our sorting network. It is based on the use of the "triplet loss", a popular surrogate for recall. We propose to apply this loss on ranks instead of similarity scores, making it only dependent on the ordering of the retrieved elements. The triplet loss on the rank can be expressed as follows:

$$\text{loss}(\mathbf{Y}[i], p, c) = \max\left\{0, \alpha + f_{\mathbf{\Theta}_B}(\mathbf{Y}[i])_p - f_{\mathbf{\Theta}_B}(\mathbf{Y}[i])_c\right\}, \tag{9}$$

where p is defined as above (the positive example in the triplet, given anchor query $\mathbf{x}_i$) and c is the index of a negative (irrelevant) example for this query. The goal is to minimize the rank of the positive pair with score $\mathbf{Y}[i]_p$ such that its rank is lower than the rank of the negative pair with score $\mathbf{Y}[i]_c$ by a margin of α.

The complete loss is then expressed over all the elements of $\mathcal{B}$ in its *hard* negative version as:

$$\mathcal{L}_{REC}(\mathbf{\Theta}_A, \mathcal{B}) = \frac{1}{d} \sum_{i \in \mathcal{B}} \max_{c \neq p, c \neq i} \text{loss}(\mathbf{Y}[i], p, c). \tag{10}$$

In equations (2), (5) and (8), the metrics are expressed in function of the non-differentiable rank function $\mathbf{rk}$. Leveraging our differentiable surrogate allows us to design a differentiable loss function for each of these metrics, respectively (4), (7) and (10).

4. Experiments

We present in this section several experiments to evaluate our approach. We first detail the way we train our differentiable sorter deep block using only synthetic data. We also present a comparison between the different models based on CNNs and on LSTM recurrent nets and with our baseline inspired from pairwise comparisons. We then evaluate the SoDeep combined with deep scoring functions $f_{\mathbf{\Theta}_B}$. The loss functions expressed in (4), (7) and (10) are applied to three different tasks: memorability prediction, cross-modal retrieval, and object recognition.

4.1. SoDeep Training and Analysis

4.1.1 Training

The proposed SoDeep models based on BI-LSTM and CNNs are trained on synthetic pairs of scores and ranks generated on the fly according to the distributions defined in Section 3.1.2.

For convenience we call an epoch as going through 100 000 pairs. The training is done using the Adam optimizer [24] with a learning rate of 0.001 which is halved every 100 epochs. Mini-batches of size 512 are used. The model is trained until the loss values stop decreasing and are stable.

Sorter model	L1 loss
Handcrafted sorter	0.0350
CNN sorter	0.0120
LSTM sorter loss	**0.0033**

Table 1: **Performance of the sorters on synthetic data**. Ranking performance of the sorter on the synthetic dataset. Among the learned sorters the LSTM one is the most efficient.

4.1.2 A handcrafted sorting baseline

We add to our trainable SoDeep blocks a baseline that does not require any training.

Inspired by the representation of the ranking problem as a matrix of pairwise ordering in [40], we build a handcrafted differentiable sorter f_h using pairwise comparisons.

A *sigmoid* function parametrized with λ scalar is used as a binary comparison function between two scalars a and b as:

$$\sigma_{comp}(a, b) = \frac{1}{1 + e^{-\lambda(b-a)}}. \tag{11}$$

Indeed, if a and b are separated by a sufficient margin, $\sigma_{comp}(a, b)$ will be either 0 or 1. The parameter λ is used to control the precision of the comparator.

This function may be used to approximate the relative rank of two components $\mathbf{y}_i$ and $\mathbf{y}_j$ in a vector $\mathbf{y}$: $\sigma_{comp}(\mathbf{y}_i, \mathbf{y}_j)$ will be close to 1 if $\mathbf{y}_i$ is (significantly) smaller than $\mathbf{y}_j$, 0 otherwise. By summing up the result of the comparison between $\mathbf{y}_i$ and all the other elements of the vector $\mathbf{y}$, we form our ranking function f_h. More precisely, the rank $f_h(\mathbf{y}, i)$ for the i-est element of $\mathbf{y}$ is expressed as follow:

$$f_h(\mathbf{y}, i) = \sum_{j : j \neq i} \sigma_{comp}(\mathbf{y}_i, \mathbf{y}_j). \tag{12}$$

The overall precision of the handcrafted sorter can be controlled by the hyper parameter λ. The value of lambda is a trade off between the precision of the predicted rank and the efficiency when back-propagating through the sorter. Further experiments will use $\lambda = 10$.

4.1.3 Results

Table 1 contains the loss values of the two different trained sorters and the handcrafted one on a generated test set of 10 000 samples. The LSTM based sorter is the most efficient, outperforming the CNN and the handcrafted sorters.

The performance of the CNN sorter slightly below the LSTM-based one can be explained by local behaviour of the CNNs, requiring a more complex structure to be able to rank elements.

In Figure 4 we compare CNN sorters with respect to their number of layers. From these results, we choose to use 8 layers in our CNN sorter since the performance seems to saturate once this depth has been reached. A possible explanation of this saturation is that the relation between the depth of the network and the input dimension ($d = 100$ here) is logarithmic.

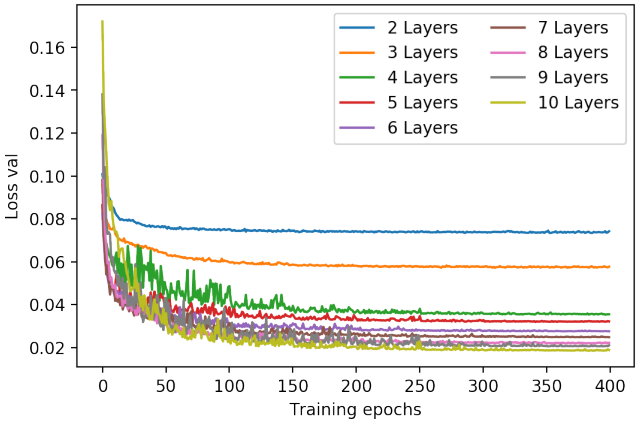


Figure 4: **Performance of the CNN sorter with respect to the depth of the CNN**. Value of the cost function during the training of multiple CNN sorters with a number of layers varying from 2 to 10. The model performances saturate for models with 8 layers or more.

4.1.4 Further analysis

The ranking function being non-continuous is non-differentiable, the rank value is jumping from one discrete value to another. We design an experiment to visualize how the different types of sorter behave at these discontinuities. Starting from a uniformly sampled vector $\mathbf{y}' \in \mathbb{R}^{100}$ of raw scores in the range $[-1, 1]$, we compute the ground truth rank $\mathbf{rk}(\mathbf{y}')_1$ and the predicted rank $f_{\Theta_B}(\mathbf{y}')_1$ of the first element y'_1 while varying this element y'_1 from -1 to 1 in increments of 0.001. The plot of the predicted ranks can be found in Fig. 5. The blue curve corresponds to the ground-truth rank where non-continuous steps are visible, whereas the curves for the learned sorters (orange and green) are a smooth approximation of the ground-truth curve.

In Fig. 6 we compare our SoDeep against previous approaches optimizing structured hinge upper bound to the mAP loss. We followed the protocol described in [36] for their synthetic data experiments. Our sorters using the loss

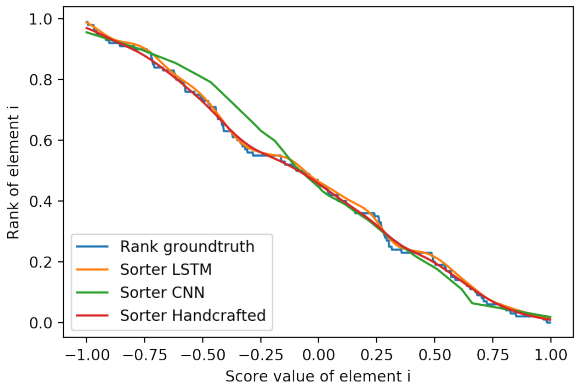


Figure 5: **Sorter behaviour analysis**. Given a synthetic vector $\mathbf{y}'$ of raw scores in the range $[-1, 1]$ of size 100 we plot the rank of its first element $\mathbf{y}'_1$ when the said value is linearly interpolated between -1 and 1. The x-axis represent the value $\mathbf{y}'_1$, and the y-axis is it corresponding rank.

$\mathcal{L}_{mAP}$ defined in (7) are compared to a re-implementation of the Hinge-AP loss proposed in [21]. The results in Fig. 6 show that our approach with the LSTM sorter (blue curve) gets mAP scores similar to [21] (purple curve) while being generic and less complex.

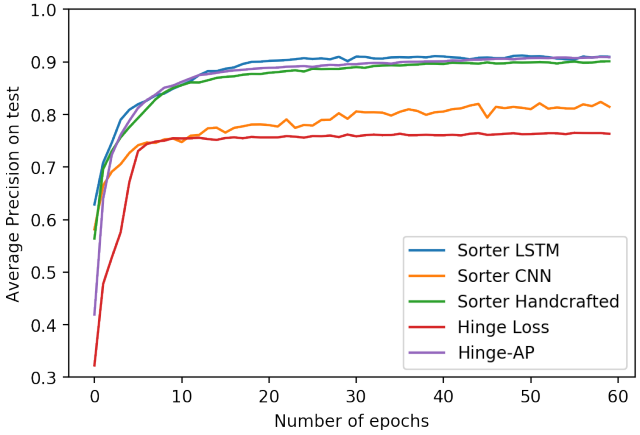


Figure 6: **Synthetic experiment on mAP optimization**. Comparison against the proposed sorter and the previous approaches.

From the learned sorters, the LSTM architecture is the one performing best on synthetic data (Tab. 1). In addition, its simple design and small number of hyper-parameters make it straightforward to train. The CNN architecture while not being as efficient, uses a smaller number of weights and is 1.7 time faster. Further experiments will use the LSTM sorter unless specified otherwise.

4.2. Differentiable Sorter based loss functions

Our method is benchmarked on three tasks. Each one of these tasks focuses on a different rank based loss function. Cross-modal retrieval will be used to test recall evaluation metrics, memorability prediction will be used for Spearman correlation and image classification will be used for mean average precision.

As explained in Section 3.1.2, a shift in distribution might appear when using sorter-based loss. To prevent this, a parallel loss can be used to help domain alignment. This loss can be used only to stabilize the initialization or kept for the whole training.

4.2.1 Spearman Correlation: Predicting Media Memorability

The media memorability prediction task [5] is used to test the differentiable sorter with respect to the Spearman correlation metrics. Examples of elements of the dataset can be found in Fig. 7. Given a 7 seconds video the task consists in predicting the short term memorability score. The memorability score reflects the probability of a video being remembered.

Figure 7: **Media memorability dataset**. Frames with low and high memorability scores coming from 4 different videos of the memorability dataset [5]. The memorability scores are overlayed on top of the images.

The task is originally on video memorability. However the model used here are pretrained on images, therefore 7 frames are extracted from each video and are associated with the memorability score of the source video. The training is done on pairs of frame and memorability score. During testing the predicted score of the 7 frames of a video are averaged to obtain the score per video. The dataset contains 8000 videos (56000 frames) for training and 2000 videos for testing. This training set is completed using LaMem dataset [23] adding 60 000 (image, memorability) pairs to the training data.

Single model	Spear. cor. test
Baseline [6]	46.0
Image only [17]	48.8
R34 + MSE loss	44.2
R34 + SoDeep loss	46.6
Sem-Emb + MSE loss	48.6
Sem-Emb + SoDeep loss	**49.4**

Table 2: **Media Memorability prediction results**. Our proposed loss function and architecture outperform the state-of-the-art system [17] by 0.6 pt.

Architectures and training The regression model consists of a feature extractor combined with a two layers MLP [33] regressing features to a single memorability score. We use two pretrained nets to extract features: the Resnet-34 [18] and the semantic embedding model of [10] (as in the next section).

We use the loss $\mathcal{L}_{SPR}$ defined in (4) to learn the memorability model. The training is done in two steps. First, for 15 epochs only the MLP layers are trained while the weights of the feature extractor are kept frozen. Second, the whole model is finetuned. The Adam optimizer [24] is used with a learning rate of 0.001 which is halved every 3 epochs. To help with domain adaptation, our loss is combined with an L1 loss for the first epoch.

Results In Tab. 2, we compare the impact of the learned loss over two architectures. For both models we defined a baseline using a L2 loss. On both architectures the proposed loss function achieves higher Spearman correlation by 2.4 points on the Resnet model and 0.8 points on the semantic embedding model. These are state of the arts result on the task with an absolute gain of 0.6 pt. The model is almost on par (-0.3 pt) with an ensemble method proposed by [17] that is using additional textual data.

Sorter comparison The memorability prediction is also used to compare the different types of sorters presented so far. Fixing the model and the hyper parameters, 4 models are trained with 4 different types of loss. The losses based on the LSTM sorter, the CNN sorter and the handcrafted sorter obtained respectively a Spearman correlation of 49.4, 46.6, 45.7, and the L1 loss gives a correlation of 46.2. These results are consistent with the result on synthetic data, with the LSTM sorter performing the best, followed by the CNN and handcrafted ones.

4.2.2 Mean Average precision: Image classification

The VOC 2007 [11] object recognition challenge is used to evaluate our sorter on a task using the mean average precision metric. We use an off-the-shelf model [9]. This model

	caption retrieval				image retrieval			
model	R@1	R@5	R@10	Med. r	R@1	R@5	R@10	Med. r
Emb. network [37]	54.9	84.0	92.2	-	43.3	76.4	87.5	-
DSVE-Loc [10]	69.8	91.9	96.6	1	55.9	86.9	94.0	1
GXN (i2t+t2i) [16]	68.5	-	**97.9**	1	**56.6**	-	**94.5**	1
DSVE-Loc + SoDeep loss	**71.5**	**92.8**	97.1	1	56.2	**87.0**	94.3	1

Table 3: **Cross-modal retrieval results on** `MS-COCO`**.** Using the proposed rank based loss function outperforms the hard negative triplet margin loss, achieving state-of-the-art results on the caption retrieval task.

Loss	mAP
VGG 16 [35]	89.3
WILDCAT [9]	95.0
WILDCAT*	93.2
WILDCAT* + SoDeep loss	94.0

Table 4: **Object recognition results**. Model marked by (*) are obtained with code available online: https://github.com/durandtibo/wildcat.pytorch

is a fully convolutional network, combining a Resnet-101 [18] with advanced spatial aggregation mechanisms.

To evaluate the loss $\mathcal{L}_{mAP}$ defined in (7) two versions of the model are trained: A baseline using only multi-label soft margin loss, and another model trained using the multi-label soft margin loss combined with $\mathcal{L}_{mAP}$.

Rows 3 and 4 of Tab. 4 show the results obtained by the two previously described models. Both models are below the state-of-the-art, however the use of the rank loss is beneficial and improves the mAP by 0.8 pt compared to the model using only the soft margin loss.

4.2.3 Recall@K: Cross-modal Retrieval

The last benchmark used to evaluate the differentiable sorter is the cross-modal retrieval. Starting from images annotated with text, we train a model producing rich features for both image and text that live in the same embedding space. Similarity in the embedding space is then used to evaluate the quality of the model on the cross-modal retrieval task.

Our approach is evaluated on the `MS-COCO` dataset [28] using the rVal split proposed in [22]. The dataset contains 110k images for training, 5k for validation and 5k for testing. Each image is annotated with 5 captions.

Given a query image (resp. a caption), the aim is to retrieve the corresponding captions (resp. image). Since `MS-COCO` contains 5 captions per image, recall at r ("R@r") for caption retrieval is computed based on whether at least one of the correct captions is among the first r retrieved ones. The task is performed 5 times on 1000-image subsets of the test set and the results are averaged.

We use an off-the-shelf model [10]. It is a two-paths multimodal embedding approach that leverages the latest neural network architecture. The visual pipeline is based on a Resnet-152 and is fully convolutional. The textual pipeline is trained from scratch and uses a Simple Recurrent Unit (SRU) [27] to encode sentences. The model is trained using the loss $\mathcal{L}_{REC}$ defined in (10) instead of the triplet based loss.

Cross-modal retrieval results can be found in Tab. 3. The model trained using the proposed loss function (DSVE-Loc + SoDeep loss) outperforms the similar architecture DSVE-Loc trained with the triplet margin based loss by (1.7%,0.9%,0.5%) on (R@1,R@5,R@10) in absolute for caption retrieval, and by (0.3%,0.1%,0.3%) for image retrieval. It obtains state-of-the-art performance on caption retrieval and is very competitive on image retrieval being almost on par with the GXN [16] model, which has a much more complex architecture. It is important to note that the loss function proposed could be beneficial for any type of architecture.

5. Conclusion

We have presented SoDeep, a novel method that leverages the expressivity of recent architectures to learn differentiable surrogate functions. Based on a direct deep network modeling of the sorting operation, such a surrogate allows us to train, in an end-to-end manner, models on a diversity of tasks that are traditionally evaluated with rank-based metrics. Remarkably, this deep proxy to estimate the rank comes at virtually no cost since it is easily trained on purely synthetic data.

Our experiments show that the proposed approach achieves very good performance on cross-modal retrieval tasks as well as on media memorability prediction and multi-label image classification. These experiments demonstrate the potential and the versatility of SoDeep. This approach allows the design of training losses that are closer than before to metrics of interest, which opens up a wide range of other applications in the future.

References

[1] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *ICML*, 2005. 2

[2] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: From pairwise approach to listwise approach. In *ICML*, 2007. 2

[3] Soumen Chakrabarti, Rajiv Khanna, Uma Sawant, and Chiru Bhattacharyya. Structured learning for non-smooth ranking losses. In *ACM SIGKDD*, 2008. 2

[4] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *J. Machine Learning Research*, 11:1109–1135, 2010. 2

[5] Romain Cohendet, Claire-Hélène Demarty, Ngoc Duong, Mats Sjöberg, Bogdan Ionescu, and Thanh-Toan Do. Mediaeval 2018: Predicting media memorability task. *arXiv preprint arXiv:1807.01052*, 2018. 2, 7

[6] Romain Cohendet, Claire-Hélène Demarty, and Ngoc Q. K. Duong. Transfer learning for video memorability prediction. In *MediaEval Workshop*, 2018. 7

[7] Yadolah Dodge. *The concise encyclopedia of statistics*. Springer Science & Business Media, 2008. 4

[8] Abhimanyu Dubey, Nikhil Naik, Devi Parikh, Ramesh Raskar, and César A Hidalgo. Deep learning the city: Quantifying urban perception at a global scale. In *ECCV*, 2016. 2

[9] Thibaut Durand, Taylor Mordan, Nicolas Thome, and Matthieu Cord. Wildcat: Weakly supervised learning of deep convnets for image classification, pointwise localization and segmentation. In *CVPR*, 2017. 2, 7, 8

[10] Martin Engilberge, Louis Chevallier, Patrick Pérez, and Matthieu Cord. Finding beans in burgers: Deep semantic-visual embedding with localization. In *CVPR*, 2018. 7, 8

[11] Mark Everingham and J Winn. The PASCAL visual object classes challenge 2007 development kit. Technical report, 2007. 2, 7

[12] Fartash Faghri, David Fleet, Jamie Ryan Kiros, and Sanja Fidler. VSE++: Improved visual-semantic embeddings. *arXiv preprint arXiv:1707.05612*, 2017. 2

[13] Basura Fernando, Efstratios Gavves, Damien Muselet, and Tinne Tuytelaars. Learning to rank based on subsequences. In *ICCV*, 2015. 2

[14] Edward H Friend. Sorting on electronic computer systems. *JACM*, 1956. 3

[15] Andrea Frome, Greg Corrado, Jon Shlens, Samy Bengio, Jeff Dean, and Tomas Mikolov. DeViSE: A deep visual-semantic embedding model. In *NIPS*, 2013. 2

[16] Jiuxiang Gu, Jianfei Cai, Shafiq Joty, Li Niu, and Gang Wang. Look, imagine and match: Improving textual-visual cross-modal retrieval with generative models. In *CVPR*, 2018. 8

[17] Rohit Gupta and Kush Motwani. Linear models for video memorability prediction using visual and semantic features. In *MediaEval Workshop*, 2018. 7

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 7, 8

[19] Alan Herschtal and Bhavani Raskutti. Optimising area under the roc curve using gradient descent. In *ICML*, 2004. 2

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 3

[21] Thorsten Joachims. Optimizing search engines using clickthrough data. In *ACM SIGKDD*, 2002. 2, 6

[22] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015. 1, 2, 8

[23] Aditya Khosla, Akhil S. Raju, Antonio Torralba, and Aude Oliva. Understanding and predicting image memorability at a large scale. In *ICCV*, 2015. 7

[24] D Kinga and J Ba Adam. A method for stochastic optimization. In *ICLR*, 2015. 5, 7

[25] Ryan Kiros, Ruslan Salakhutdinov, and Richard Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014. 1, 2

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[27] Tao Lei and Yu Zhang. Training RNNs as fast as CNNs. *arXiv preprint arXiv:1709.02755*, 2017. 8

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 8

[29] David Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999. 1

[30] Lin Ma, Zhengdong Lu, Lifeng Shang, and Hang Li. Multimodal convolutional neural networks for matching image and sentence. In *ICCV*, 2015. 2

[31] Pritish Mohapatra, Michal Rolinek, CV Jawahar, Vladimir Kolmogorov, and M Kumar. Efficient optimization for rank-based loss functions. In *CVPR*, 2018. 2

[32] Mehryrar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. MIT Press, 2012. 2

[33] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 1958. 7

[34] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 1997. 3

[35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 8

[36] Yang Song, Alexander Schwing, and Raquel Urtasun. Training deep neural networks via direct loss minimization. In *ICML*, 2016. 2, 6

[37] Liwei Wang, Yin Li, Jing Huang, and Svetlana Lazebnik. Learning two-branch neural networks for image-text matching tasks. *IEEE Trans. Pattern Recognition and Machine Intell.*, 41(2):394–407, 2018. 1, 8

[38] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *J. Machine Learning Research*, 2009. 2

[39] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *NIPS*, 2003. 2

[40] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *ACM SIGIR*, 2007. 2, 4, 5