INDUCER SELECTION PRINCIPLES FOR DEEPFUSION SYSTEMS

Mihai Gabriel CONSTANTIN[1], Liviu-Daniel ȘTEFAN[2], Bogdan IONESCU[3]

The current landscape of ensemble learning or late fusion approaches is dominated by methods that employ a very low number of inducer systems, while using traditional approaches with regards to the fusion engine, predominantly statistical, weighted, Bagging or Random Forests. Even with the advent of deep learning, few approaches use deep neural networks in building the ensemble decision and improving the results of single-system approaches. One of these methods is represented by the DeepFusion set of approaches, that integrate a very large number of inducer systems, while providing significantly improved final performance over the performance of its component inducers. However, no attempt has yet been made for DeepFusion with regards to reducing and optimizing the set of inducers, while maintaining the same level of performance. Thus, this paper proposes a set of methods for inducer selection and reduction, based on their performance and on their similarity computed via clustering. Our methods are tested on the popular Interestingness10k dataset, that provides data and inducers for the prediction of image and video visual interestingness. We present an in-depth analysis of the performance of the optimization methods, with regards to the results according to the main performance metric associated with this dataset, as well as the degree to which these methods reduce the number of utilized inducers.

Keywords: DeepFusion, inducer selection, late fusion, ensembling, optimization

1. Introduction

Despite the present advancements in information retrieval, single learners do not perform well when working with data involving multipartite entanglement including concept drift, noisy data, class imbalance, high-dimensionality, etc. In this context, ensemble learning tries to fill this gap by exploiting a set of machine learning algorithms through the combination of their individual predictions. Here, ensemble learning is a general term for approaches that create predictions using a pool of inducers, often in supervised machine learning problems. Inducers are algorithms that map input instances to categories via a consensus mechanism, encapsulated in a model (e.g., a classifier or regressor),

[1] University Politehnica of Bucharest, Romania, e-mail: `mihai.constantin84@upb.ro`

[2] University Politehnica of Bucharest, Romania

[3] University Politehnica of Bucharest, Romania

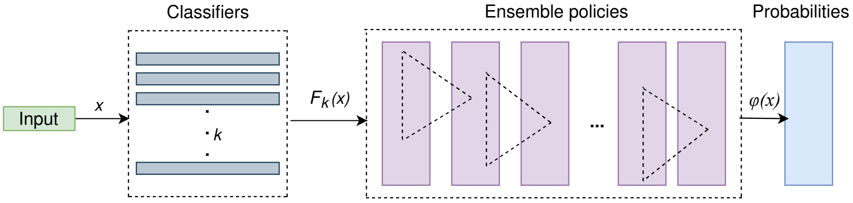


Figure 1. Illustration of a general ensembling framework. Given a sample x, ensembling aims to expand the space of representable functions, by combining the hypotheses F. The goal is to find the optimal one, meaning that it should approximate ground truth as close as possible to reduce the generalization error, in contrast to single classifiers, where the target is to find the optimal hypothesis in the given space H.

all tailored to the type of inputs being examined. Given a probability on a hypothesis, the consensus methods may rely on majority voting, weighting or statistical inference, or more intricate approaches, such as the use of hierarchical or network architectures. A general ensemble architecture is illustrated in Figure 1.

An unsolved research question is how to choose the right collection of inducers for a classification problem in order to produce an accurate ensemble. Therefore, many factors must be considered when selecting the best ensemble setting. One of the aspects to be considered is the selection of the inducers that will be incorporated in the ensemble architecture. A good practice is to construct an architecture with mutually complementary classifiers characterized by *high diversity*, although it is not guaranteed that boosting ensemble diversity will increase its performance in practice [19]. Here diversity refers to the variations in decisions or predictions output by the inducers when analyzing the same instance. One hypothesis is to consider the correlations between accuracy and diversity since ensemble predictive capability is constrained by the rule of large numbers. This paradigm is supported by the no-free-lunch theorem formulated by Wolpert [30], which states that *"for any two learning algorithms there are just as many situations appropriately weighted in which algorithm one is superior to algorithm two as vice versa according to any of the measures of superiority."* The best scenario is when the individual classifiers of an ensemble are entirely complementary. In contrast, the worst scenario implies that all the individual classifiers are identical, in both positive and negative predictions.

For efficient learning in a wide range of applications [23, 22, 17], deep neural networks (DNNs) have taken over as the de facto standard. One of the major results of state-of-the-art deep learning architectures [16, 28] is that they can detect subtle structures in large data sets because to their better representation expertise for high-dimensional data, and when combined with their classification abilities, they significantly outperform traditional descriptors and classifiers. While ensemble learning has a fairly long history in machine learning, there is currently little literature on ensemble learning with DNNs.

The remainder of the paper proceeds as follows. We first position our work in the literature, discussing related approaches and concepts in Section 2. Furthermore, the proposed optimization schemes are presented in Section 3, while the experimental setup and results are presented in Section 4. The paper concludes with Section 5.

2. Previous work

In the literature, multiple terms have been employed to denote a set of functions that collaborate to resolve a machine learning problem, such as *aggregation*, *committee*, *late fusion* and *ensembles* [18, 12, 3]. *Ensemble learning* is a methodology that employs a pool of learners to generate ensemble prediction by combining the members' predictions via a consensus method. Instead of relying on a single robust classifier, ensembling advocates improving system accuracy by combining multiple classifiers, as summarized in [5]. In a traditional setting, ensemble methods comprise two phases: the generation of predictive models, and the learning process. The former describes how the classifiers are built, and the latter specifies how the classifiers are organized within the ensemble and how their predictions are used to form the overall ensemble prediction. An additional intermediate phase called ensemble pruning [20, 2] is also proposed in the literature, responsible for the selection of the inducers prior to combination. Popular boosting methods like AdaBoost [13] with it's variants, e.g., soft margin AdaBoost [26], Gradient boosting machines [14], and XGBoost [7], represent one of the goto approaches when creating ensembles. Other notable approaches include Bagging [4], Random Forests [6], and Extremely randomized trees codes [15].

Despite DNNs' recent success, there is limited research on the incorporation of ensemble learning in deep neural networks. One of the first pure ensembling approaches use DNNs is represented by the DeepFusion set of methods [27, 8]. While these networks bring significant improvements with over single-system predictors, no study has yet been performed with regards to optimizing and reducing the set of inducers they employ. A study in this direction would allow for comparable or even better system performance, while reducing the computational need of these systems by reducing the number of inducers.

In this context, the contribution of this work is as follows: (i) we propose a set of three inducer selection methods that would allow for optimization based on inducer performance and clustering; (ii) we test these methods on the Interestingness10k dataset, comparing their performance with the performance of the original DeepFusion approaches; (iii) we analyze the results, while taking into account the gain in performance compared with how efficient the selection method is with regards to the reduced set of inducers.

3. Proposed approach

Our approach starts from the DeepFusion [9] ensembling methods, representing a continuation of our experiments on the prediction of media interestingness, using inducer data gathered during the MediaEval benchmarking competition[1], corresponding to the Interestingness10K dataset [10]. We explore different methods of inducer selection, with the main target of attaining comparable or better results compared with our previous experiments, while considerably reducing the number of inducers used in the experiments. This would address one of the problems often cited with regards to ensemble building - the considerable cost and system requirements necessary for training and especially running a large number of inducers.

In theory, given the system described in Figure 1, the proposed approaches involve pruning the set of k inducers. We can thus reduce the input space for the ensemble policies from the $F_k(x)$ to a new $F_r(x)$ set of hypothesis. This aspect is presented in Figure 2. Thus, given the initial space of hypothesis, $H(k)$, the role of the reduction function $R(H(k))$ is to provide a new set of inducers $H(r)$, so that the new set is a subset of the original one: $H(r) \subset H(k)$. Therefore, no transformation is applied to the inducer functions themselves, as some of them are just taken out of the final decision and are no longer used as inducers. Thus, the new $H(r)$ set of inducers is a subset of $H(k)$. We propose several methods to achieve this optimization, as follows:

- Performance-based selection - where inducers that perform below a certain threshold are dropped;
- Middle-out-based selection - where inducers between certain thresholds are dropped;
- Cluster-based selection - where certain representatives of clusters of similar inducers are dropped.

3.1. Performance-based selection

Given the $H(k)$ set of inducers, and the R optimization function, we propose selecting only those inducers that perform above a certain threshold σ, where the performance of each inducer is computed using a function Γ. We theorize that, by dropping inducers that perform badly, we may be able to help the network learn from better performing inducers, thus increasing its final results. Thus, the performance of each individual inducer can be computed as $\Gamma(H(i))_{i=0}^{k}$. The condition for keeping or dropping a certain inducer and the final set of optimized inducers $H(r)$ can be expressed as $H(r) = \{H(i)|0 \leqslant i < k, \Gamma(H(i)) \geqslant \sigma\}$.

[1] `https://multimediaeval.github.io/`

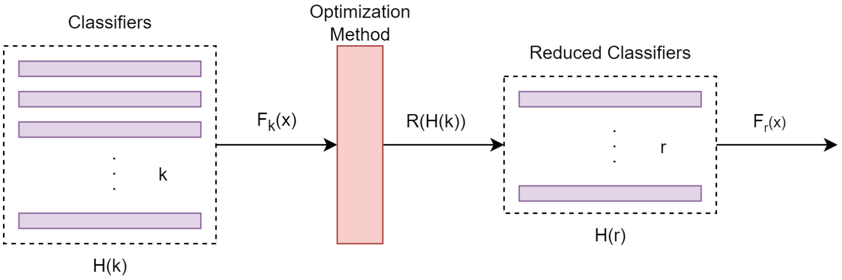


Figure 2. Illustration of a general optimization method for ensemble classification. Given k classifiers that produce the output $F_k(x)$, the role of the optimization method is to create a function R that will select a subset of inducers and will only run the inference on that subset. Thus, the input to the Ensemble Policies will consist of a reduced $F_r(x)$ set.

3.2. Middle-out-based selection

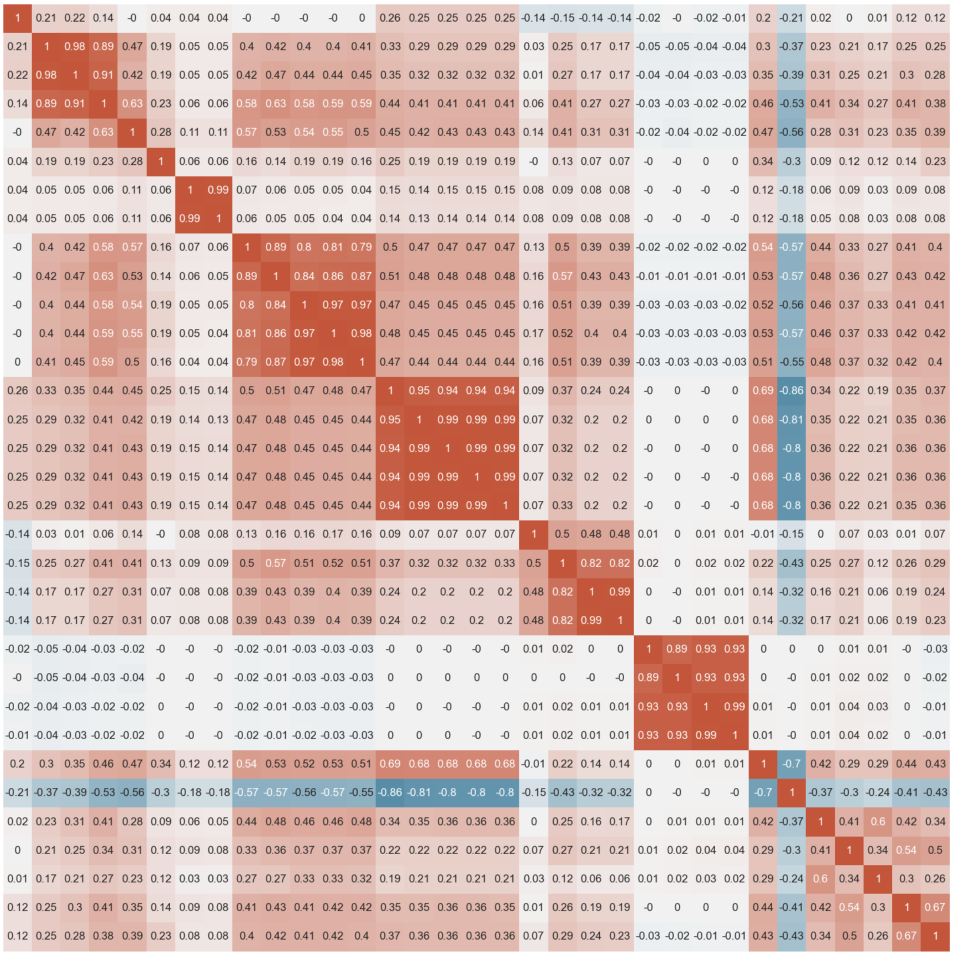
Thresholding is again the main focus of the second type of inducer selection. However, this time, we theorize that the fusion system may be able to learn even from inducers that perform badly. Theoretically, inducers that are below a lower threshold α may positively help the learning process of the DeepFusion system, serving as negative examples in the training phase. Therefore, we propose only dropping inducers with performances between two thresholds, intuitively the inducers that perform neither well nor badly. We call these two thresholds α (the lower one) and β (the higher one), and the set of optimized inducers can be expressed as $H(r) = \{H(i) | 0 \leqslant i < k, \alpha \leqslant \Gamma(H(i)) \leqslant \beta\}$.

3.3. Cluster-based selection

For the final optimization method, we theorize that clusters of inducers may naturally be created, as different classifiers may converge towards the similar conclusions regarding the data. We thus create these clusters by using a distance metric between two sets of predictions i and j as denoted as $\Gamma_{i,j}$. The natural formation of clusters can be visualized in Figure 3, where a similarity matrix between all the possible pairs of inducers is presented. We used a hierarchical Aglomerative Clustering [21] method for computing the clusters of inducers. Using an average scheme for creating the clusters, the measure of dissimilarity between two clusters X and Y can be expressed as:

$$d(X,Y) = \frac{1}{|X|\,|Y|} \sum_{x \in X} \sum_{y \in Y} \Gamma(x,y) \tag{1}$$

Given this distance, a maximum distance $dmax$ can be used as the valid distance for cluster creation at training time, so that clusters can only be created if $d(X,Y) \leqslant dmax$ An update decision regarding the joining of two



1	0.21	0.22	0.14	-0	0.04	0.04	0.04	-0	-0	-0	-0	0	0.26	0.25	0.25	0.25	0.25	-0.14	-0.15	-0.14	-0.14	-0.02	-0	-0.02	-0.01	0.2	-0.21	0.02	0	0.01	0.12	0.12
0.21	1	0.98	0.89	0.47	0.19	0.05	0.05	0.4	0.42	0.4	0.4	0.41	0.33	0.29	0.29	0.29	0.29	0.03	0.25	0.17	0.17	-0.05	-0.05	-0.04	-0.04	0.3	-0.37	0.23	0.21	0.17	0.25	0.25
0.22	0.98	1	0.91	0.42	0.19	0.05	0.05	0.42	0.47	0.44	0.44	0.45	0.35	0.32	0.32	0.32	0.32	0.01	0.27	0.17	0.17	-0.04	-0.04	-0.03	-0.03	0.35	-0.39	0.31	0.25	0.21	0.3	0.28
0.14	0.89	0.91	1	0.63	0.23	0.06	0.06	0.58	0.63	0.58	0.59	0.59	0.44	0.41	0.41	0.41	0.41	0.06	0.41	0.27	0.27	-0.03	-0.03	-0.02	-0.02	0.46	-0.53	0.41	0.34	0.27	0.41	0.38
-0	0.47	0.42	0.63	1	0.28	0.11	0.11	0.57	0.53	0.54	0.55	0.5	0.45	0.42	0.43	0.43	0.43	0.14	0.41	0.31	0.31	-0.02	-0.04	-0.02	-0.02	0.47	-0.56	0.28	0.31	0.23	0.35	0.39
0.04	0.19	0.19	0.23	0.28	1	0.06	0.06	0.16	0.14	0.19	0.19	0.16	0.25	0.19	0.19	0.19	0.19	-0	0.13	0.07	0.07	-0	-0	0	0	0.34	-0.3	0.09	0.12	0.12	0.14	0.23
0.04	0.05	0.05	0.06	0.11	0.06	1	0.99	0.07	0.06	0.05	0.05	0.04	0.15	0.14	0.15	0.15	0.15	0.08	0.09	0.08	0.08	-0	-0	-0	-0	0.12	-0.18	0.06	0.09	0.03	0.09	0.08
0.04	0.05	0.05	0.06	0.11	0.06	0.99	1	0.06	0.05	0.05	0.04	0.04	0.14	0.13	0.14	0.14	0.14	0.08	0.09	0.08	0.08	-0	-0	-0	-0	0.12	-0.18	0.05	0.08	0.03	0.08	0.08
-0	0.4	0.42	0.58	0.57	0.16	0.07	0.06	1	0.89	0.8	0.81	0.79	0.5	0.47	0.47	0.47	0.47	0.13	0.5	0.39	0.39	-0.02	-0.02	-0.02	-0.02	0.54	-0.57	0.44	0.33	0.27	0.41	0.4
-0	0.42	0.47	0.63	0.53	0.14	0.06	0.05	0.89	1	0.84	0.86	0.87	0.51	0.48	0.48	0.48	0.48	0.16	0.57	0.43	0.43	-0.01	-0.01	-0.01	-0.01	0.53	-0.57	0.48	0.36	0.27	0.43	0.42
-0	0.4	0.44	0.58	0.54	0.19	0.05	0.05	0.8	0.84	1	0.97	0.97	0.47	0.45	0.45	0.45	0.45	0.16	0.51	0.39	0.39	-0.03	-0.03	-0.03	-0.02	0.52	-0.56	0.46	0.37	0.33	0.41	0.41
-0	0.4	0.44	0.59	0.55	0.19	0.05	0.04	0.81	0.86	0.97	1	0.98	0.48	0.45	0.45	0.45	0.45	0.17	0.52	0.4	0.4	-0.03	-0.03	-0.03	-0.03	0.53	-0.57	0.46	0.37	0.33	0.42	0.42
0	0.41	0.45	0.59	0.5	0.16	0.04	0.04	0.79	0.87	0.97	0.98	1	0.47	0.44	0.44	0.44	0.44	0.16	0.51	0.39	0.39	-0.03	-0.03	-0.03	-0.03	0.51	-0.55	0.48	0.37	0.32	0.42	0.4
0.26	0.33	0.35	0.44	0.45	0.25	0.15	0.14	0.5	0.51	0.47	0.48	0.47	1	0.95	0.94	0.94	0.94	0.09	0.37	0.24	0.24	-0	0	-0	0	0.69	-0.86	0.34	0.22	0.19	0.35	0.37
0.25	0.29	0.32	0.41	0.42	0.19	0.14	0.13	0.47	0.48	0.45	0.45	0.44	0.95	1	0.99	0.99	0.99	0.07	0.32	0.2	0.2	0	0	0	0	0.68	-0.81	0.35	0.22	0.21	0.35	0.36
0.25	0.29	0.32	0.41	0.43	0.19	0.15	0.14	0.47	0.48	0.45	0.45	0.44	0.94	0.99	1	0.99	0.99	0.07	0.32	0.2	0.2	-0	0	-0	0	0.68	-0.8	0.36	0.22	0.21	0.36	0.36
0.25	0.29	0.32	0.41	0.43	0.19	0.15	0.14	0.47	0.48	0.45	0.45	0.44	0.94	0.99	0.99	1	0.99	0.07	0.32	0.2	0.2	-0	0	-0	-0	0.68	-0.8	0.36	0.22	0.21	0.36	0.36
0.25	0.29	0.32	0.41	0.43	0.19	0.15	0.14	0.47	0.48	0.45	0.45	0.44	0.94	0.99	0.99	0.99	1	0.07	0.33	0.2	0.2	-0	0	-0	-0	0.68	-0.8	0.36	0.22	0.21	0.35	0.36
-0.14	0.03	0.01	0.06	0.14	-0	0.08	0.08	0.13	0.16	0.16	0.17	0.16	0.09	0.07	0.07	0.07	0.07	1	0.5	0.48	0.48	0.01	0	0.01	0.01	-0.01	-0.15	0	0.07	0.03	0.01	0.07
-0.15	0.25	0.27	0.41	0.41	0.13	0.09	0.09	0.5	0.57	0.51	0.52	0.51	0.37	0.32	0.32	0.32	0.33	0.5	1	0.82	0.82	0.02	0	0.02	0.02	0.22	-0.43	0.25	0.27	0.12	0.26	0.29
-0.14	0.17	0.17	0.27	0.31	0.07	0.08	0.08	0.39	0.43	0.39	0.4	0.39	0.24	0.2	0.2	0.2	0.2	0.48	0.82	1	0.99	0	-0	0.01	0.01	0.14	-0.32	0.16	0.21	0.06	0.19	0.24
-0.14	0.17	0.17	0.27	0.31	0.07	0.08	0.08	0.39	0.43	0.39	0.4	0.39	0.24	0.2	0.2	0.2	0.2	0.48	0.82	0.99	1	0	-0	0.01	0.01	0.14	-0.32	0.17	0.21	0.06	0.19	0.23
-0.02	-0.05	-0.04	-0.03	-0.02	-0	-0	-0	-0.02	-0.01	-0.03	-0.03	-0.03	-0	0	-0	-0	-0	0.01	0.02	0	0	1	0.89	0.93	0.93	0	0	0	0.01	0.01	-0	-0.03
-0	-0.05	-0.04	-0.03	-0.04	-0	-0	-0	-0.02	-0.01	-0.03	-0.03	-0.03	0	0	0	0	0	0	0	-0	-0	0.89	1	0.93	0.93	0	-0	0.01	0.02	0.02	0	-0.02
-0.02	-0.04	-0.03	-0.02	-0.02	0	-0	-0	-0.02	-0.01	-0.03	-0.03	-0.03	-0	0	-0	-0	-0	0.01	0.02	0.01	0.01	0.93	0.93	1	0.99	0.01	-0	0.01	0.04	0.03	0	-0.01
-0.01	-0.04	-0.03	-0.02	-0.02	0	-0	-0	-0.02	-0.01	-0.02	-0.03	-0.03	0	0	0	-0	-0	0.01	0.02	0.01	0.01	0.93	0.93	0.99	1	0.01	-0	0.01	0.04	0.02	0	-0.01
0.2	0.3	0.35	0.46	0.47	0.34	0.12	0.12	0.54	0.53	0.52	0.53	0.51	0.69	0.68	0.68	0.68	0.68	-0.01	0.22	0.14	0.14	0	0	0.01	0.01	1	-0.7	0.42	0.29	0.29	0.44	0.43
-0.21	-0.37	-0.39	-0.53	-0.56	-0.3	-0.18	-0.18	-0.57	-0.57	-0.56	-0.57	-0.55	-0.86	-0.81	-0.8	-0.8	-0.8	-0.15	-0.43	-0.32	-0.32	0	-0	-0	-0	-0.7	1	-0.37	-0.3	-0.24	-0.41	-0.43
0.02	0.23	0.31	0.41	0.28	0.09	0.06	0.05	0.44	0.48	0.46	0.46	0.48	0.34	0.35	0.36	0.36	0.36	0	0.25	0.16	0.17	0	0.01	0.01	0.01	0.42	-0.37	1	0.41	0.6	0.42	0.34
0	0.21	0.25	0.34	0.31	0.12	0.09	0.08	0.33	0.36	0.37	0.37	0.37	0.22	0.22	0.22	0.22	0.22	0.07	0.27	0.21	0.21	0.01	0.02	0.04	0.04	0.29	-0.3	0.41	1	0.34	0.54	0.5
0.01	0.17	0.21	0.27	0.23	0.12	0.03	0.03	0.27	0.27	0.33	0.33	0.32	0.19	0.21	0.21	0.21	0.21	0.03	0.12	0.06	0.06	0.01	0.02	0.03	0.02	0.29	-0.24	0.6	0.34	1	0.3	0.26
0.12	0.25	0.3	0.41	0.35	0.14	0.09	0.08	0.41	0.43	0.41	0.42	0.42	0.35	0.35	0.36	0.36	0.35	0.01	0.26	0.19	0.19	-0	0	0	0	0.44	-0.41	0.42	0.54	0.3	1	0.67
0.12	0.25	0.28	0.38	0.39	0.23	0.08	0.08	0.4	0.42	0.41	0.42	0.4	0.37	0.36	0.36	0.36	0.36	0.07	0.29	0.24	0.23	-0.03	-0.02	-0.01	-0.01	0.43	-0.43	0.34	0.5	0.26	0.67	1

Figure 3. Similarity matrix between different runs. Simmilar runs ($r > 0$) are presented in red, while dissimilar ones ($r < 0$) are presented in blue. We can observe the formation of several clusters, generally around runs submitted by the same teams.

clusters X and Y, with n_X and n_Y elements in them, can be expressed as a distance comparison to any other cluster Z:

$$\frac{n_X d(X,Z) + n_Y d(Y,Z)}{n_X + n_Y} \tag{2}$$

In the final step we drop the worst performing inducer from each cluster containing more than one inducer. We theorize that this type of approach should reduce network input redundancy to a certain point, while directly targeting and dropping the lowest performing inducer from certain important clusters.

4. **Experimental results**

The following section will present details regarding the data used, the inducers and their sources and the results of the three selection methods.

4.1. **Experimental data**

We propose a set of experiments derived from those presented in [9]. Concretely, we use the version of Interestingness10k associated with the MediaEval 2017 Predicting Media Interestingness task [11]. 9,831 photos and videos make up the dataset, including 2,435 samples in the testing set (testset) and 7,396 samples in the development set (devset). The benchmarking competition required participants to create and train their media interestingness prediction algorithms on the devset and run the systems on samples from the testset. We target both the image and video subtasks, and use the systems and predictions submitted by the participants as inducers for our DeepFusion system. A high number of systems are available: 33 inducers for the image subtask and 42 for the video subtask.

We select a RSKF75 approach, as defined in [9]: we split the inducer predictions according to a Random Stratified K-Fold approach, where 75% of the data is kept for training, and 25% of the data is used for testing the results of the proposed approaches. We also select the same baseline systems for comparing our performance with the current literature, namely: the system with the top performance during the MediaEval competition, the system with the top performance from the current literature on the Interestingness10k dataset, and a set of state-of-the-art traditional fusion methods. We will ultimately compare the results of the optimized DeepFusion approaches with the un-optimized versions of DeepFusion.

4.2. **Results**

The results of our experiments are presented in Table 1, where the three selection schemes are presented as follows: performance-based selection is represented as O1, middle-out-based selection as O2, and cluster-based selection as O3. It is encouraging to note that, while O2 has low levels of success, O1 and O3 represent better approaches across both image and video prediction. Consequently, we propose that middle-out optimization does not represent a good additions for the inducer ensemble. It is also interesting to note that the most successful approach varies from task to task. For the image task, the most successful approach is the performance-based selection approach, resulting in a MAP@10 value of 0.3568. On the other hand, for the video task, the most successful approach is the cluster-based selection method, with a MAP@10 value of 0.2891. This may be an effect of the data and the inducers themselves, and while there is no general conclusion with regards to the better performer between O1 and O3, we theorize that, for future tasks and implementations of

Table 1. Optimization results are compared with baseline runs published in the literature (b), results from the best performing traditional fusion methods, namely AdaBoost (a), the results of the DeepFusion experiments (d), as well as the results of the proposed optimization schemes (p). The three selection schemes are represented as O1 (performance-based selection), O2 (middlebased selection), and O3 (cluster-based selection).

	Image		Video	
Type	System	MAP@10	System	MAP@10
(b)	[25]	0.1385	[1]	0.0827
	[24]	0.1985	[29]	0.093
(a)	[13]	0.1674	[13]	0.1129
(d)	Dense	0.3355	Dense	0.2677
	Attn	0.3389	Attn	0.2750
	Conv	0.3436	Conv	0.2799
	CSF	0.3403	CSF	0.2825
(p)	**O1**	**0.3568**	O1	0.2711
	O2	0.3119	O2	0.2303
	O3	0.3428	**O3**	**0.2891**

this system, both approaches must be tested and implemented in order to select the most appropriate approach. The improvement over the un-optimized DeepFusion approaches is approximately 3.84% for O1 in the image prediction subtask, while for the video subtask, O3 optimizer improves the performance by 2.33%. These performances are achieved with a lower number of inducers, reducing the computational demand, as we will show in the following section.

4.3. Analysis

We perform a performance analysis on the three proposed methods, altering the parameters used for selecting the inducers (i.e., σ for performance-based selection, the (α, β) pair for middle-out-based selection) and creating the clusters (i.e. the $dmax$ distance for cluster-based selection). Figure 4 presents the results of this series of experiments. Values for the parameters were chosen empirically, mainly because parameters have different effects depending on the task they are applied to. For example, while experiments with a σ value of 0.09 can be performed on the Image task, this value would be too high for the Video task as it would drop almost all the inducers. Values tested for the (α, β) pairs for middle-out optimization are as follows: $\{(0.07, 0.09), (0.065, 0.09), (0.0650.095), (0.06, 0.095)\}$ for the Image task, and $\{(0.052, 0.059), (0.05, 0.06), (0.05, 0.065), (0.048, 0.067)\}$ for the Video task. Results show a decrease in performance when compared with the base DeepFusion Dense approach of 7.03% for the Image task and of 13.9% for the Video task.

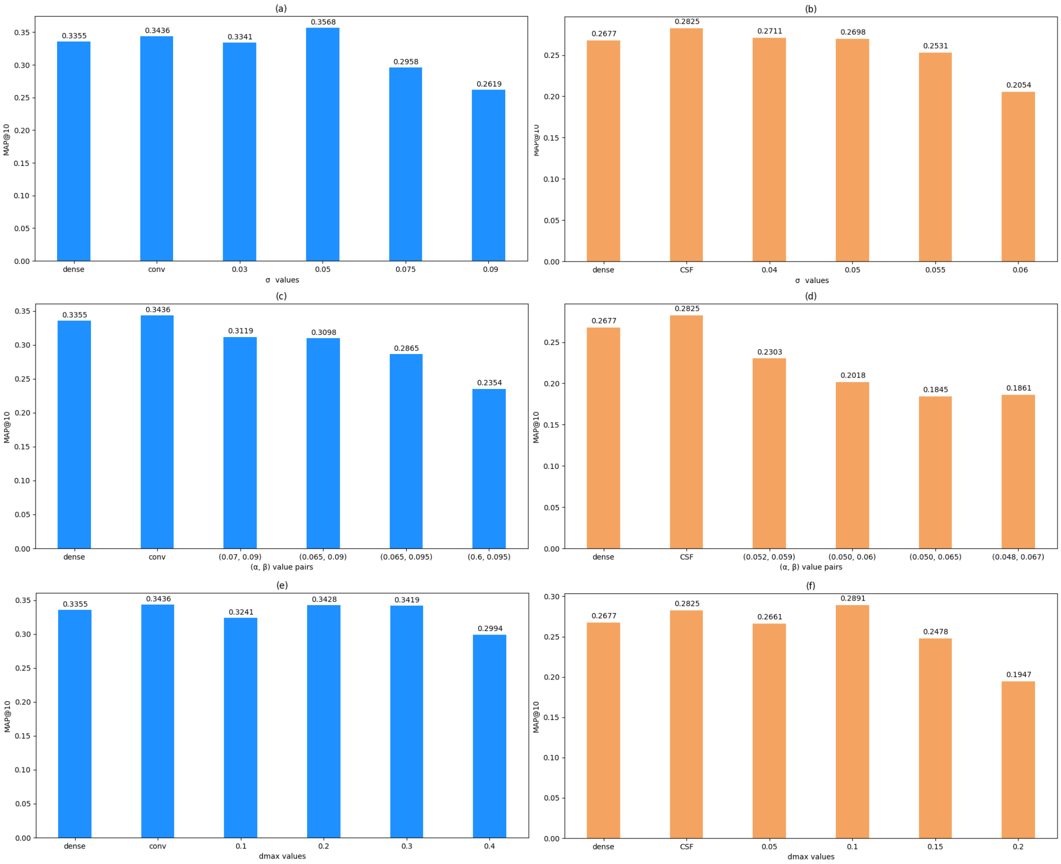


FIGURE 4. Result analysis for the three optimization methods with regards to their parameters. We present results on the Image (in blue – a, c, e) and Video (in brown – b, d, f) tasks, while varying the σ parameter for O1 (a, b), the α and β parameters for O2 (c, d) and $dmax$ for O3 (e, f).

With regards to the performance-based optimization schemes, we propose the following σ values: $\{0.03, 0.05, 0.075, 0.09\}$ for O1 scheme on the Image task, and $\{0.04, 0.05, 0.055, 0.06\}$ on the Video task respectively. The best result for O1 on the Image task is achieved with the parameter $\sigma = 0.05$ and represents a value of $MAP@10 = 0.3568$, a 6.3% increase over the basic DeepFusion Dense approach and 3.84% increase over the DeepFusion Convolutional approach, the best performing result with the un-optimized version of DeepFusion. With the help of this scheme, the number of inducers was decreased from 33 to 28. On the other hand, for the Video task, the best results are attained with $\sigma = 0.04$, reducing the number of inducers from 42 to 40, and reaching a performance of 0.2711. This represents a 1.27% increase over the un-optimized Dense approach and a decrease of 4.03% decrease when compared with the CSF approach.

Finally, the cluster-based selection method uses the following *dmax* values for the Image task: $\{0.1, 0.2, 0.3, 0.4\}$ and for the Video task: $\{0.05, 0.1, 0.15, 0.2\}$. The best result for the O3 scheme on the Image task is achieved with a *dmax* value of 0.2 reaching a MAP@10 value of 0.3428, representing a 2.17% increase over the DeepFusion Dense approach, and a decrease of 0.23% compared with the Convolutional approach, reducing the number of inducers from 33 to 27. For the Video task, the best results are obtained with a *dmax* value of 0.1, scoring a MAP@10 value of 0.2891. This scheme reduces the number of inducers from 42 to 36, while it increases performance by 7.99% over the Dense DeepFusion approach, and 2.33 % over the CSF approach.

5. Conclusions

This work presents the creation and performance of several optimization schemes for DeepFusion approaches. We propose three optimization schemes, with the target of reducing the number of inducers used by the system, while maintaining comparable, or even increased performance. The following approaches were proposed: (i) performance-based selection, where the worst performing inducers are dropped; (ii) middle-out-based selection, where inducers between certain thresholds are dropped; (iii) the cluster-based selection, where the worst performing representatives of inducer clusters are dropped.

Results are tested on the Interestingness10k dataset, composed of an Image and a Video prediction task, while the inducers associated with this dataset are used as inputs for the proposed systems. In our experiments we show that the performance- and cluster-based methods both lower the necessary inducers by dropping them according to each optimization scheme's principles, while even managing in some cases to have better results. We theorize that higher performance is most likely the result of lowering the noise the bad inducers create in the input space.

We also propose that optimization schemes, for DeepFusion approaches in particular, and for all kinds of late fusion approaches in general, represent an interesting and worthwhile direction of study, reducing the computational and cost impact of implementing such approaches. While different schemes and parameters represented the top performing approach in the Image and Video prediction scenarios, we believe that other sets of inducers and data may behave differently, favoring either of these two schemes, while parameters would have to be adapted to each new task.

Acknowledgements

The work of M.G. Constantin has been funded by the Ministry of Investments and European Projects through the Human Capital Sectoral Operational Program 2014-2020, Contract no. 62461/03.06.2022, SMIS code 153735. The work of L.-D. Ștefan and B. Ionescu has been funded by the Ministry of

Research, Innovation, and Digitization UEFISCDI, projects IDENEO, “Identifying the Unusual Behavior of People in Video Streams”, Solutions Axis, grant 29SOL/2021, PN-III-P2-2.1-SOL-2021-0024 and DeepVisionRomania, “Identifying People in Video Streams using Silhouette Biometrics”, Solutions Axis, grant 28SOL/2021, PN-III-P2-2.1-SOL-2021-0114.

REFERENCES

[1] Olfa Ben-Ahmed et al. “Eurecom@ mediaeval 2017: Media genre inference for predicting media interestingnes”. In: MediaEval 2017 Workshop. 2017.

[2] Yijun Bian et al. “Ensemble Pruning Based on Objection Maximization With a General Distributed Framework”. In: IEEE transactions on neural networks and learning systems (2019).

[3] Christopher M Bishop et al. Neural networks for pattern recognition. Oxford university press, 1995.

[4] Leo Breiman. “Bagging predictors”. In: Machine learning 24.2 (1996).

[5] Leo Breiman. Bias, variance, and arcing classifiers. Tech. rep. Tech. Rep. 460, Statistics Department, University of California, Berkeley, 1996.

[6] Leo Breiman. “Random forests”. In: Machine learning 45.1 (2001), pp. 5–32.

[7] Tianqi Chen and Carlos Guestrin. “Xgboost: A scalable tree boosting system”. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM. 2016, pp. 785–794.

[8] Mihai Gabriel Constantin, Liviu-Daniel Ştefan, and Bogdan Ionescu. “DeepFusion: deep ensembles for domain independent system fusion”. In: MultiMedia Modeling: 27th International Conference, MMM 2021, Prague, Czech Republic, June 22–24, 2021, Proceedings, Part I 27. Springer. 2021, pp. 240–252.

[9] Mihai Gabriel Constantin, Liviu-Daniel Ştefan, and Bogdan Ionescu. “Exploring deep fusion ensembling for automatic visual interestingness prediction”. In: Human Perception of Visual Information: Psychological and Computational Perspectives (2022), pp. 33–58.

[10] Mihai Gabriel Constantin et al. “Visual interestingness prediction: A benchmark framework and literature review”. In: International Journal of Computer Vision 129 (2021), pp. 1526–1550.

[11] Claire-Hélène Demarty et al. “Mediaeval 2017 predicting media interestingness task”. In: MediaEval 2017 Workshop. 2017.

[12] Harris Drucker et al. “Boosting and other ensemble methods”. In: Neural Computation 6.6 (1994), pp. 1289–1301.

[13] Yoav Freund, Robert Schapire, and Naoki Abe. “A short introduction to boosting”. In: Journal-Japanese Society For Artificial Intelligence 14 (1999).

[14] Jerome H Friedman. “Greedy function approximation: a gradient boosting machine”. In: Annals of statistics (2001), pp. 1189–1232.

[15] Pierre Geurts, Damien Ernst, and Louis Wehenkel. “Extremely randomized trees”. In: Machine learning 63.1 (2006), pp. 3–42.

[16] Kaiming He et al. “Deep residual learning for image recognition”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, pp. 770–778.

[17] Tero Karras et al. “Progressive growing of gans for improved quality, stability, and variation”. In: arXiv preprint arXiv:1710.10196 (2017).

[18] Josef Kittler et al. “On combining classifiers”. In: IEEE transactions on pattern analysis and machine intelligence 20.3 (1998), pp. 226–239.

[19] Ludmila I Kuncheva and Christopher J Whitaker. “Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy”. In: Machine learning 51.2 (2003), pp. 181–207.

[20] Gonzalo Martínez-Muñoz, Daniel Hernández-Lobato, and Alberto Suárez. “An analysis of ensemble pruning techniques based on ordered aggregation”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 31.2 (2008), pp. 245–259.

[21] Daniel Müllner. “Modern hierarchical, agglomerative clustering algorithms”. In: arXiv preprint arXiv:1109.2378 (2011).

[22] Anh Nguyen et al. “Plug & play generative networks: Conditional iterative generation of images in latent space”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017, pp. 4467–4477.

[23] Aaron van den Oord et al. “Wavenet: A generative model for raw audio”. In: arXiv preprint (2016).

[24] Jayneel Parekh, Harshvardhan Tibrewal, and Sanjeel Parekh. “Deep pairwise classification and ranking for predicting media interestingness”. In: Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval. 2018.

[25] Reza Aditya Permadi et al. “DUT-MMSR at MediaEval 2017: Predicting Media Interestingness Task.” In: MediaEval 2017 Workshop. 2017.

[26] Gunnar Rätsch, Takashi Onoda, and K-R Müller. “Soft margins for AdaBoost”. In: Machine learning 42.3 (2001), pp. 287–320.

[27] Liviu-Daniel Ştefan, Mihai Gabriel Constantin, and Bogdan Ionescu. “System fusion with deep ensembles”. In: Proceedings of the 2020 International Conference on Multimedia Retrieval. 2020, pp. 256–260.

[28] Hugo Touvron et al. “Fixing the train-test resolution discrepancy”. In: Advances in Neural Information Processing Systems. 2019, pp. 8250–8260.

[29] Shuai Wang et al. “Video interestingness prediction based on ranking model”. In: Proceedings of the joint workshop of the 4th workshop on affective social multimedia computing and first multi-modal affective computing of large-scale multimedia data. 2018, pp. 55–61.

[30] David H Wolpert. “The supervised learning no-free-lunch theorems”. In: Soft computing and industry. Springer, 2002, pp. 25–42.