Proc. 26th Int. Conf. on Multimedia Modeling (MMM 2020), Springer LNCS vol. 11961. Author's accepted version.
The final publication is available at Springer via https://doi.org/10.1007/978-3-030-37731-1_19

Subclass deep neural networks: re-enabling neglected classes in deep network training for multimedia classification

Nikolaos Gkalelis[0000−0001−6741−3334] and Vasileios Mezaris[0000−0002−0121−4364]

CERTH-ITI, 6th Km Charilaou-Thermi Road,
P.O. BOX 60361 Thessaloniki, Greece
{gkalelis, bmezaris}@iti.gr

Abstract. During minibatch gradient-based optimization, the contribution of observations to the updating of the deep neural network's (DNN's) weights for enhancing the discrimination of certain classes can be small, despite the fact that these classes may still have a large generalization error. This happens, for instance, due to overfitting, i.e. to classes whose error in the training set is negligible, or simply when the contributions of the misclassified observations to the updating of the weights associated with these classes cancel out. To alleviate this problem, a new criterion for identifying the so-called "neglected" classes during the training of DNNs, i.e. the classes which stop to optimize early in the training procedure, is proposed. Moreover, based on this criterion a novel cost function is proposed, that extends the cross-entropy loss using subclass partitions for boosting the generalization performance of the neglected classes. In this way, the network is guided to emphasize the extraction of features that are discriminant for the classes that are prone to being neglected during the optimization procedure. The proposed framework can be easily applied to improve the performance of various DNN architectures. Experiments on several publicly available benchmarks including, the large-scale YouTube-8M (YT8M) video dataset, show the efficacy of the proposed method[1].

Keywords: DNN · neglected classes · subclasses · cross-entropy loss.

1 Introduction and related work

Deep neural networks (DNNs) have shown a breakthrough performance in many machine learning problems and are currently witnessing a significant commercial deployment in several application domains such as multimedia understanding, self-driving cars, IoT and other. The state-of-the-art DNNs for classification tasks consist of a series of weight layers, nonlinear activation functions and downsampling operators and on top of them an output layer typically equipped with a

[1] Source code is made publicly available at: `https://github.com/bmezaris/subclass_deep_neural_networks`

sigmoid or softmax activation function modeling c categorical probability distributions [16,21]. An important aspect on the design of a DNN is the choice of the cost function and the optimization algorithm. The cross-entropy (CE) loss and the stochastic gradient descent (SGD) combined with the back-propagation (BP) algorithm for updating the DNN parameters are almost always the sole choice in practice [12]. The great success of those DNNs is based on their extraordinary ability to extract nonlinear features at different layers guided by the SGD-BP algorithm in order to transform a set of c (possibly) nonlinear classification tasks in the input space of the DNN to c linear ones in the input space of the output layer. More specifically, for the ith output node a gradient update to the correct direction is generated, whose length is proportional to the training error of the ith class, guiding the overall network to extract the desired features and producing a linearly separable subspace for the ith classification task. In [19], it is shown that the application of the CE loss with gradient descent on separable data convergences to the max-margin solution with a logarithmic convergence rate. Moreover, it is shown that the above analysis is also valid in deep networks if after a certain number of iterations the weight vectors of the last weight layer are assumed fixed and the class distributions at its output are considered linearly separable (or piecewise linearly separable). However, as we show in this paper not all weight vectors in the last layers yield a linearly separable problem simultaneously and thus not all class separating hyperplanes converge to the max-margin solution with the same rate. Instead, there is an antagonism, where the extraction of discriminant features for certain classes is emphasized during the optimization of the DNN, while other classes are partially neglected yielding a "less" linearly separable problem in the input space of the output layer for these classes, and thus a separating margin that is suboptimal.

The limitation of DNNs to treat all classes fairly during the training procedure has been mostly studied in the context of class imbalanced learning [15]. Moreover, the identification of classes receiving little attention during training as described above is a relatively unexplored topic. To this end, a new criterion for identifying such neglected classes is proposed. This criterion computes the contribution of positive and negative observations in the gradient update of the weight vectors in the output layer and combines the computed quantities to form a stable measure for the likelihood that the underlying class is going to be neglected. Moreover, in order to turn the attention of the DNN on the identified neglected classes, we resort to a subclass partitioning strategy. Subclass-based classification techniques have been successfully used in the shallow learning paradigm. In [10], learning vector quantization (LVQ) is used to find a set of cluster centers for each class and classification is performed by finding the closest class center. In [7], mixture discriminant analysis (MDA) fits a Gaussian mixture density to each class, extending the linear discriminant analysis (LDA) to the non-normal setting. In [5], nonlinear classification problems are solved by splitting the original set of classes to subclasses and embedding the binary problems in a problem-dependent subclass error-correcting output codes (SECOC) design. In [20,6], a set of kernel subclass discriminant analysis techniques are proposed in order to deal with nonlinearly separable subclasses, and it is shown that the identification of the optimum kernel parameters can be performed more easily exploiting the subclass partitions.

Motivated by the above works, a subclass DNN (SDNN) framework is proposed, where the neglected classes are augmented and partitioned to subclasses, and subsequently a novel subclass CE (SCE) loss, which emphasizes the separation of subclasses belonging to different classes, is applied to train the network. In this way, the network is trained to derive a piecewise linear subspace for the neglected classes, imposing a less strict requirement for the extraction of nonlinear features for these classes. Thus, the DNN is trained more effectively with respect to the neglected classes, increasing its overall generalization performance. The novel SDNN framework is compared with state-of-the-art approaches in 3 popular benchmarks (CIFAR10, CIFAR100 [11] and SVHN [14]) and in the large-scale YT8M video dataset [1] for the task of multiclass and multilabel classification, respectively. The results show that in most cases the proposed SDNNs obtain significant performance improvements.

The rest of the paper is structured as follows: Section 2 presents the proposed method and Section 3 describes the experimental evaluation. Conclusions are drawn in Section 4.

2 Proposed method

2.1 Identification of neglected classes

Suppose a DNN with a sigmoid output layer (SG)

$$\mathbf{h}_\kappa = \mathbf{W}^T \mathbf{x}_\kappa + \mathbf{b}, \tag{1}$$

$$q_{i,\kappa} = \frac{1}{1 + \exp(-h_{i,\kappa})}, \tag{2}$$

where $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_c] \in \mathbb{R}^{f \times c}$, $\mathbf{b} = [b_1, \dots, b_c]^T$ are the weight matrix and bias vector of the SG layer, and c is the number of classes. Moreover, assuming a batch of n training observations, the vectors $\mathbf{x}_\kappa = [x_{1,\kappa}, \dots, x_{f,\kappa}]^T$, $\mathbf{h}_\kappa = [h_{1,\kappa}, \dots, h_{c,\kappa}]^T$, $\mathbf{q}_\kappa = [q_{1,\kappa}, \dots, q_{c,\kappa}]^T$, $\mathbf{y}_\kappa = [y_{1,\kappa}, \dots, y_{c,\kappa}]^T$, are associated with the κth training observation in the batch, and are the input and output vector of the linear part of the SG layer, the output vector of the SG layer, and the class indicator vector, respectively. The ith component of $\mathbf{y}_\kappa$ is the label of the κth observation with respect to the ith class, i.e. $y_{i,\kappa}$ equals one if $\mathbf{x}_\kappa \in \omega_i$ and zero otherwise, and ω_i denotes the ith class. For training the DNN, the minibatch stochastic gradient descent (SGD) and the CE loss are used

$$L = -\frac{1}{n} \sum_{\kappa=1}^{n} \sum_{i=1}^{c} (y_{i,\kappa} \ln(q_{i,\kappa}) + (1 - y_{i,\kappa}) \ln(1 - q_{i,\kappa})). \tag{3}$$

Under this framework, the weight vector associated with the ith class is updated at each iteration as below

$$\mathbf{w}_i = \mathbf{w}_i - \eta \mathbf{g}_i, \tag{4}$$

where, $\mathbf{g}_i = \frac{1}{n}\sum_{\kappa=1}^{n}\zeta_{i,\kappa}\mathbf{x}_\kappa$ is the gradient of L with respect to $\mathbf{w}_i$, η is the learning rate and $\zeta_{i,\kappa} = q_{i,\kappa} - y_{i,\kappa}$. Noting that $q_{i,\kappa} \in [0,1]$ we observe that $\zeta_{i,\kappa} \in [-1,1]$, with $\zeta_{i,\kappa} \approx 0$ when the right answer for $\mathbf{x}_\kappa$'s label is provided by SG layer's unit i, and $\zeta_{i,\kappa}$ moving towards $|1|$ as the likelihood of unit i to provide a wrong answer increases

$$\zeta_{i,\kappa} = \begin{cases} 1 \text{ if } & q_{i,\kappa} = 1, y_{i,\kappa} = 0, \\ -1 \text{ if } & q_{i,\kappa} = 0, y_{i,\kappa} = 1, \\ 0 \text{ if } & q_{i,\kappa} == y_{i,\kappa}. \end{cases} \tag{5}$$

These properties of $\zeta_{i,\kappa}$ can aid the correct operation of the gradient-based learning approach, i.e., shrinking the gradient in (4) when the right answer is obtained, and providing a strong gradient otherwise, forcing the overall network to act quickly in order to correct the mislabeled observations. However, this is not always the case. For instance, considering that the contribution to the summand in (4) of different observations may cancel out, the gradient may shrink despite the fact that many observations are misclassified. To see this, we rewrite the gradient as

$$\mathbf{g}_i = \frac{1}{n}(\tilde{\boldsymbol{\delta}}_i - \hat{\boldsymbol{\delta}}_i) = \frac{1}{n}\boldsymbol{\delta}_i, \tag{6}$$

$$\hat{\boldsymbol{\delta}}_i = \sum_{\mathbf{x}_\kappa \in \omega_i} -\zeta_{i,\kappa}\mathbf{x}_\kappa, \tag{7}$$

$$\tilde{\boldsymbol{\delta}}_i = \sum_{\mathbf{x}_\kappa \notin \omega_i} \zeta_{i,\kappa}\mathbf{x}_\kappa, \tag{8}$$

where $\hat{\boldsymbol{\delta}}_i$, $\tilde{\boldsymbol{\delta}}_i$ equal zero when the positive and negative observations, respectively, are classified correctly. Note that $-\zeta_{i,\kappa}, \mathbf{x}_\kappa \in \omega_i$ and $\zeta_{i,\kappa}, \mathbf{x}_\kappa \notin \omega_i$ are less than one and always positive, and thus $\hat{\boldsymbol{\delta}}_i$, $\tilde{\boldsymbol{\delta}}_i$ are the weighted means of the target and non-target class, respectively, weighted with the likelihood derived from the DNN that this observation belongs to the respective category or not. When $\hat{\boldsymbol{\delta}}_i$, $\tilde{\boldsymbol{\delta}}_i$ are close to each other, the overall gradient $\boldsymbol{\delta}_i$ approaches zero and $\mathbf{w}_i$ remains relatively unchanged, despite the fact that many observations are still not classified correctly by unit i. When this undesired effect appears, the network gradually stops to optimize the weights of the different layers below for extracting discriminant features associated with such "neglected" classes, paying more attention on improving the training classification rates of classes which still produce a strong gradient at each iteration. A unit i with large $\|\hat{\boldsymbol{\delta}}_i\|$, $\|\tilde{\boldsymbol{\delta}}_i\|$ and at the same time small difference between these two quantities reflects a high likelihood that the associated class is not getting the required attention and is going to be neglected in subsequent iterations. Based on the analysis above, every τ minibatch iterations we compute the following measure for estimating how likely a class is to be neglected

$$\theta_i = \frac{1}{n\tau}\sum_{l=p-\tau+1}^{\tau}\frac{\|\hat{\boldsymbol{\delta}}_{i,l}\| + \|\tilde{\boldsymbol{\delta}}_{i,l}\|}{\|\boldsymbol{\delta}_{i,l}\|}, \tag{9}$$

where $\hat{\boldsymbol{\delta}}_{i,l}$, $\tilde{\boldsymbol{\delta}}_{i,l}$ are the gradient terms (7), (8) at the lth minibatch iteration, $\|\|$ is the vector norm operator and p is the current iteration. The identification of the most neglected class $\imath$ is then performed by using a simple argmax rule

$$\imath = \underset{i}{\operatorname{argmax}}(\theta_i). \tag{10}$$

2.2 SDNNs

The major consequence of neglecting a class during the optimization procedure is that the trained DNN will fail to learn an appropriate feature mapping where the neglected classes are linearly separable. To alleviate this unwanted behavior we propose the use a clustering algorithm to derive a subclass partition for those classes that are prone to be neglected. By exploiting this partition it is expected that it will be generally easier for the DNN to learn a nonlinear mapping where the subclasses are linearly separable. Under this framework, the easiest way to extend the CE criterion would be to treat each subclass as a class. However, this loss will treat equivalently the costs associated with misclassifying an observation to the non-target subclasses without examining which non-target subclasses are associated with the target class of the observation and which not. To this end, we propose the following loss in order to favor the separability of those subclasses that correspond to different classes

$$L = -\frac{1}{n}\sum_{\kappa=1}^{n}\sum_{i=1}^{c}\sum_{j=1}^{H_i}(y_{i,j,\kappa}\ln(q_{i,j,\kappa}) + (1 - y_{i,\kappa})\ln(1 - q_{i,j,\kappa})), \tag{11}$$

where, $y_{i,j,\kappa}$ is the label of the κth training observation in the batch associated with jth subclass of class i, i.e., $y_{i,j,\kappa}$ equals one if $\mathbf{x}_\kappa \in \omega_{i,j}$ and zero otherwise, and $h_{i,j,\kappa}$, $q_{i,j,\kappa}$ are the input and output to the activation function of the (i,j) unit associated with $\mathbf{x}_\kappa$. Note, that in the second summand of (11) the class label $y_{i,\kappa}$ is utilized instead of the subclass label $y_{i,j,\kappa}$ in order to emphasize the separation of subclasses belonging to different classes, as explained above.

2.3 Subclass partitioning and augmentation

Any clustering algorithm and augmentation approach can be applied to derive a subclass division of the neglected classes. However, for large-scale datasets such as the YT8M [1], it may be infeasible to use computationally demanding clustering approaches such as k-means. To this end, the lightweight approach described in Algorithm 1 for partitioning the observations of the ith class into two subclasses is proposed. It is based on the computation of the distance of each class observation to $\mathbf{m}$, which is the mean along all observations in the training set and used as a representation of the rest-of-world class. Moreover, data augmentation can be performed to the neglected classes by applying extrapolation in the feature space for each observation as proposed in [3]

$$\acute{\mathbf{x}}_{i,j,\kappa} = \lambda(\mathbf{x}_{i,j,\kappa} - \check{\mathbf{x}}_{i,j}) + \mathbf{x}_{i,j,\kappa}, \tag{12}$$

Algorithm 1 Subclass partitioning algorithm

```
Input: $\{\mathbf{x}_{i,1}, \dots, \mathbf{x}_{i,n_i}\}$, $\mathbf{m}$
Output: $\mathbf{x}_{i,j,\kappa}$, $\check{\mathbf{x}}_{i,j}$,   $\kappa = 1, \dots, n_{i,j}$, $j = 1, 2$
1: Compute $d_j = \|\mathbf{x}_{i,j} - \mathbf{m}\|\ \forall j$
2: Sort $\mathbf{x}_{i,j}$'s in descending order according to the $d_j$'s: $\{\tilde{\mathbf{x}}_{i,1}, \dots, \tilde{\mathbf{x}}_{i,n_i}\}$
3: Compute $n_{i,1} = \lfloor n_i/2 \rfloor$, $n_{i,2} = n_i - n_{i,1}$
4: Set $\{\mathbf{x}_{i,1,1}, \dots, \mathbf{x}_{i,1,n_{i,1}}\} = \{\tilde{\mathbf{x}}_{i,1}, \dots, \tilde{\mathbf{x}}_{i,n_{i,1}}\}$ (i.e., we assign the observations with
   the highest $d_j$'s to the 1st subclass; the remaining observations go to the 2nd
   subclass as shown in step 5)
5: Set $\{\mathbf{x}_{i,2,1}, \dots, \mathbf{x}_{i,2,n_{i,2}}\} = \{\tilde{\mathbf{x}}_{i,n_{i,1}+1}, \dots, \tilde{\mathbf{x}}_{i,n_i}\}$
6: Set $\check{\mathbf{x}}_{i,1} = \tilde{\mathbf{x}}_{i,1}$, $\check{\mathbf{x}}_{i,2} = \tilde{\mathbf{x}}_{i,n_i}$ (these quantities are used in (12) for the augmentation)
```

where, $\lambda \in [0, 1]$ and $\check{\mathbf{x}}_{i,1}$, $\check{\mathbf{x}}_{i,2}$ are the observations of class i with the largest and smallest distance from $\mathbf{m}$, respectively. Using the approach described in this section, both class partitioning and augmentation can be performed very efficiently on-line without the need to load the whole dataset or large parts of it in memory.

3 Experiments

3.1 Validation of the neglection criterion

In order to verify the validity of the proposed criterion we train and evaluate a VGG16 network for 420 epochs in the CIFAR10 dataset and record the testing CCR$_i$, the neglection measure θ_i, and the gradient vectors $\boldsymbol{\delta}_i$, $\hat{\boldsymbol{\delta}}_i$ and $\tilde{\boldsymbol{\delta}}_i$ for each epoch and class i, $i = 0, \dots, 9$. The exact details of the network architecture and the training procedure are provided in Section 3.2. The recorded values for θ_i and CCR$_i$ are shown in Figure 1, while the length of the three gradients plotted between the epochs 100 and 200 are depicted in Figure 2. We observe the following: i) There is a clear correlation between the generalization error rate and the neglection criterion. More specifically, as shown in Figure 1 the neglection values can be used to rank the classes in terms of their expected generalization

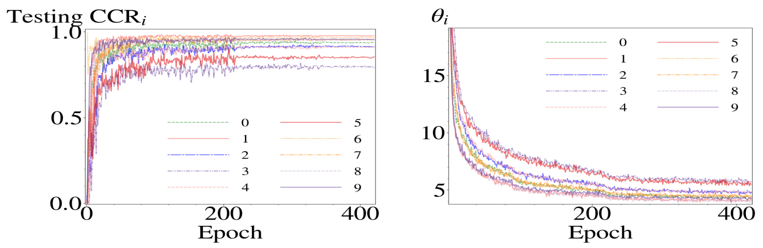

Fig. 1. Testing CCR$_i$ and neglection measure θ_i for the CIFAR10 dataset.

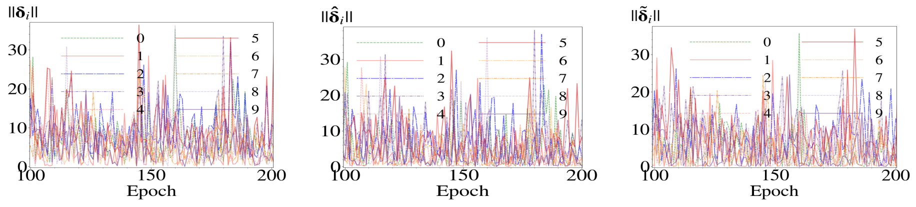

Fig. 2. Length of gradient vectors $\boldsymbol{\delta}_i$, $\hat{\boldsymbol{\delta}}_i$ and $\tilde{\boldsymbol{\delta}}_i$ for the CIFAR10 dataset.

performance. ii) From the CCR rates, the classes can be roughly categorized into two groups, i.e., one group with classes 3 and 5 that attain a rather low CCR and another group with the rest of the classes having better CCR rates. Looking at the 0 to 30 epoch temporal segment we observe that the classes of the first group clearly exhibit a smaller rate of CCR increase, while the majority of the ones in the second group almost attain their steady-state condition during this period. Moreover, after the 10th epoch a CCR gap between the first and second group of more than 10% in absolute values is observed, which stabilizes after the 230th epoch. Exactly the same conclusions can be drawn from the evolution of the θ_i values, where in this case a gap of 1 unit between the two groups is observed after the 30th epoch. iii) The norm of the gradient update $\|\boldsymbol{\delta}_i\|$ alone, or its contributing parts $\|\hat{\boldsymbol{\delta}}_i\|$, $\|\tilde{\boldsymbol{\delta}}_i\|$, exhibit high fluctuations and a rather noisy behavior, and their direct observation does not provide any valuable information concerning the generalization performance of the classes during the training procedure.

From the above analysis we can see that a group of classes is neglected during the optimization procedure and that the proposed criterion can be used to identify these classes, verifying the theoretical analysis in Section 2.1.

3.2 Multiclass classification using SDNNs

Datasets For the experimental evaluation of the proposed approach in the problem of multiclass classification the following 3 datasets are used: i) The CIFAR-10 and CIFAR-100 datasets [11] consist of 60000 32×32 color images each, drawn from 10 and 100 classes, respectively. Both datasets are divided to a training and test partition with 50000 and 10000 images respectively. ii) The street view house numbers (SVHN) dataset [14] contains 630420 color images of 32×32 pixel resolution, similar to the CIFAR datasets. They depict house numbers extracted from Google Street View images, i.e., each image belongs to one of ten classes. The dataset is split to a training, testing and an extra partition of 73257, 26032 and 531131 images, respectively. Following the standard procedure for this dataset, the training and extra partitions are combined in our experiments to form a new training partition.

Experimental setup Two modern DNN architectures are used for the evaluation of the proposed approach, namely, the VGG16 [16] with batch normalization after every convolutional layer, and two variants of the wide residual networks (WRN) [21] depending on the dataset. Specifically, a WRN with depth 28, widening factor 10 (WRN-28-10) and dropout rate of 0.4 is used for the CIFAR datasets, and the WRN-16-8 with 0.3 dropout rate is employed for the SVHN dataset. The reason that these two WRN architectures are employed is because they have exhibited state-of-the-art performance in the above datasets [4]. All networks are trained for 200 epochs using the CE loss (3), minibatch SGD with Nesterov momentum of 0.9, batch size of 128, weight decay of 0.0005, and an exponential learning rate schedule set to decrease at the 60th, 120th and 160th epoch. For the CIFAR datasets, the initial learning rate is set to 0.1 and reduced by a factor of 0.1 according to the learning rate schedule above, while for the SVHN dataset an initial learning rate and reduction factor of 0.01 and 0.2 are used, respectively. The images are normalized per-channel to zero mean and unit variance, and data augmentation is performed during training following [4], i.e., 4 pixels zero-padding and random cropping, horizontal mirroring with 50% probability, and cutout 16×16 and 8×8 for the CIFAR-10, CIFAR-100 datasets, respectively. The SVHN undergoes the same normalization, however, only 20×20 cutout is used to augment this dataset.

The subclass VGG16 (SVGG16) and WRN (SWRN) are created as explained in the following. The original VGG16 and WRN are executed for 30 epochs in order to compute a reliable neglection score θ_i for each class. In this way, 2 classes from the CIFAR10 and SVHN (20% of the total classes) and 10 classes from the CIFAR100 (10% of the total classes) with the highest θ_i's are selected, i.e., the classes with labels 3, 5 from CIFAR10, 1, 3 from SVHN and 0, 11, 18, 35, 53, 55, 62, 69, 72, 88 from CIFAR100. In order to alleviate any class imbalance problems resulting from the partitioning to subclasses, the selected classes are first doubled in size using the augmentation method described in [9], and then the k-means algorithm is applied to create two new subclasses from each class. The augmented datasets are then used to train SVGG and SWRN using the SCE loss (11) and the training procedure described above for the conventional networks. Learning is performed using the training partition of the datasets and the performance of each method is measured using the correct classification rate (CCR) along all classes achieved by the trained network in the test set.

All networks are implemented in PyTorch, extending the code provided in [4,21], and the experimental evaluation is performed in an Intel i7 3770K@3.5Ghz PC with 32 GB RAM, Windows 10, and Nvidia GeForce GPU (GTX 1080 Ti).

Results The evaluation results in terms of CCR and training times in hours are shown in Table 1. The testing times are only a few seconds in all cases (spanning the range of 5 secs for VGG16 in CIFAR10 to 12 secs for SWRN in SVHN). From the obtained results we can see that the proposed SVGG16 and SWRN outperform the conventional networks in all datasets, with differences in performance from 0.21% (SWRN over WRN in SVHN) to $\approx 2.5\%$ (SVGG16

Table 1. Accuracy rates and training times (hours) in 3 datasets

	VGG16 [16]	*SVGG16*	*WRN [4]*	*SWRN*
CIFAR10	93.5% (2.6 h)	94.8% (2.7 h)	96.92% (7.1 h)	**97.14%** (8.1 h)
CIFAR100	71.24% (2.6 h)	73.67% (2.6 h)	81.59% (7.1 h)	**82.17%** (7.5 h)
SVHN	98.16% (29.1 h)	98.35% (34.1 h)	98.70% (33.1 h)	**98.81%** (42.7 h)

over VGG16 in CIFAR100). Considering that the CCR rates obtained with the WRN combined with cutout regularization [4] are currently among the state-of-the-art performances, even the small improvements obtained with the proposed approach are considered significant. Moreover, we observe that the training time overhead caused by the application of the subclass approach is negligible for the medium size CIFAR datasets, and relatively small for the much larger SVHN dataset.

3.3 Multilabel classification using SDNNs

Dataset The YT8M [1] is utilized to evaluate the proposed approach for the task of multilabel classification. This is the largest publicly available multilabel video dataset consisting of 6134598 videos annotated with one or more labels from 3862 classes (3.4 labels per video on average). For facilitating the comparison of different classification techniques the dataset is already divided to a training, evaluation and testing partition, consisting of 3888919, 1112356 and 1133323 videos, respectively. Visual and audio feature vectors in $\mathbb{R}^{1024}$ and $\mathbb{R}^{128}$, respectively, are already provided at video-level as well as at frame-level granularity. The data is stored in Tensorflow's tfrecord file format (3844 shards for each data partition and granularity level), which offers very efficient import and preprocessing functionalities for large-scale datasets.

Experimental setup For the evaluation, a rather simple convolutional neural network (CNN) is utilized with a convolutional, a max-pooling, a dropout and a SG layer of c outputs. The convolutional layer consists of 64 one-dimensional (1D) filters and is equipped with a rectification (ReLU) nonlinearity. Each filter has a receptive field of size 3 and stride 1, and zero padding is applied in order to preserve the spatial size of the input signals. The max-pooling layer employs a filter of size 2 and stride 2, while a keep-rate of 0.7 is used for the dropout layer. The CE loss (3) combined with the minibatch SGD-BP algorithm and weight decay of 0.0005 is used for training the CNN. The training is performed over 5 epochs with an exponential learning rate schedule, initial learning rate of 0.001, learning rate decay 0.95 in every epoch, and batch size of 512.

For the construction of the subclass CNN (SCNN), the CNN above is initially applied in the training set for $\frac{1}{3}$ of an epoch in order to obtain a neglection value θ_i (9) for each YT8M class and the 386 classes with the highest θ_i are selected, i.e., 10% of the total number of classes. The selected classes are then partitioned to $H_i = 2$ subclasses using the efficient on-line algorithm described in Algorithm 1, avoiding the loading of the whole dataset or large parts of it in memory, which would be infeasible for the YT8M dataset. Moreover, data augmentation is performed to the neglected classes using the extrapolation technique described in Section 2.3, setting $\lambda = 0.5$. In this way the number of observations in each subclass partition is doubled. The resulting SCNN is trained using the proposed SCE loss (11) and the training procedure described for the conventional CNN. For completeness, a standard logistic regression (LR) classifier is also evaluated using the same training procedure with initial learning rate of 0.001.

We performed experiments with the video-level visual features, as well as with audio-visual features produced by concatenating the video-level visual and audio feature vectors. In all cases L2-normalization was applied. The models are trained and evaluated using the YT8M training and validation set respectively. The labeling information for the testing set is not provided and for this reason it is excluded from the evaluation. Nevertheless, as reported in relevant works [8] the performance difference on the validation and test set is negligible. The evaluation metrics of the YT8M Video Understanding Challenge [1] are used to report our results, namely, Hit@1, precision at equal recall rate (PERR), mean average precision (mAP), and global average precision at 20 (GAP@20), with the latter being the official metric of the YT8M challenge for ranking the different participating teams. The models are implemented in Tensorflow and the evaluation is performed in the same PC used in Section 3.2.

Table 2. Evaluation results in YT8M

	Visual			*Visual + Audio*		
	LR	*CNN*	*SCNN*	*LR*	*CNN*	*SCNN*
Hit@1	82.4%	82.5%	**83.2%**	82.3%	85.2%	**85.7%**
PERR	71.9%	72.2%	**72.9%**	71.8%	75.4%	**75.9%**
mAP	41.2%	42.3%	**45.2%**	40.1%	45.6%	**47.9%**
GAP@20	77.1%	77.6%	**78.6%**	77%	80.7%	**82.2%**
T_{tr} *(min)*	**18.7**	59.2	66.2	**18.9**	60.3	67.1

Table 3. Comparison with the best single-model approaches in YT8M

	[8]	*[13]*	*[2]*	*[18]*	*SCNN*
GAP@20	82.15%	80.9%	**82.5%**	82.25%	82.2%

Results The evaluation results in terms of Hit@1, PERR, mAP, GAP@20 and training time (T_{tr}) in minutes for each method are shown in Table 2. Moreover, in

Table 3 we show state-of-the-art results achieved from single-model approaches in YT8M. From the analysis of the obtained results we observe the following: i) The SCNN attains the best results, outperforming the conventional CNN by 1% and 1.5% GAP using the visual and audio-visual features, respectively. Both networks outperform the standard LR. ii) By exploiting the audio information both CNN and SCNN attain a significant performance gain of more than 3%. On the other hand, a degradation in performance is observed for the LR model, which most likely does not have the capacity to exploit the additional discriminant information provided by the audio modality. iii) As shown in Table 3, our SCNN method achieving a GAP of 82.2% performs in par with the best single-model approaches reported in [8,13,2,18]. This is an excellent performance considering that our SCNN exploits only the video-level feature vectors provided by the YT8M dataset in contrast to the top-performers in the competition, which additionally exploit the frame-level visual features and build upon stronger and much more computationally-demanding feature vector descriptors such as Fisher Vectors, VLAD, BoW, and other [2,18]. We should also note that the best performing approach [17] in the YT8M competition achieved a GAP score of 88.9%. However, this is achieved using an ensemble of classifiers and a variety of feature descriptors (e.g. NetVLAD, FVNet, DBoF), whose extraction and use would increase the computation requirements by at least an order of magnitude; thus this approach cannot be fairly compared with our proposed approach that creates a single model using the video-level descriptors already provided in the YT8M dataset.

4 Conclusions

In this paper, a novel SDNN framework was proposed and evaluated in two different multimedia classification tasks. Firstly, a new criterion is used for the identification of neglected classes during the initial stages of the DNN training. Subsequently, the identified classes are partitioned to subclasses and augmented in order to formulate an easier, piecewise linear classification problem for the DNN, and a new cross-entropy loss function emphasizing the discrimination of subclasses belonging to different classes is utilized. In this way, SDNNs are forced to pay more attention to the neglected classes, increasing effectively the overall classification performance. The experimental evaluation in two different problem domains, specifically, multiclass classification in three popular benchmarks (CIFAR10, CIFAR100 and SVHN) and multilabel classification using the YT8M dataset, which is the largest publicly available dataset for this task, demonstrated the efficacy of the proposed approach.

Acknowledgments

This work was supported by the EUs Horizon 2020 research and innovation programme under grant agreement H2020-780656 ReTV.

References

1. Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, A.P., Toderici, G., Varadarajan, B., Vijayanarasimhan, S.: YouTube-8M: A large-scale video classification benchmark. In: arXiv:1609.08675 (2016)
2. Bober-Irizar, M., Husain, S., Ong, E.J., Bober, M.: Cultivating DNN diversity for large scale video labelling. In: CVPR Workshops (2017)
3. DeVries, T., Taylor, G.W.: Dataset augmentation in feature space. In: ICLR Workshops. Toulon, France (Apr 2017)
4. Devries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. arXiv:1708.04552 (2017)
5. Escalera, S. et al.: Subclass problem-dependent design for error-correcting output codes. IEEE Trans. Pattern Anal. Mach. Intell. **30**(6), 1041–1054 (Jun 2008)
6. Gkalelis, N., Mezaris, V., Kompatsiaris, I., Stathaki, T.: Mixture subclass discriminant analysis link to restricted Gaussian model and other generalizations. IEEE Trans. Neural Netw. Learn. Syst. **24**(1), 8–21 (Jan 2013)
7. Hastie, T., Tibshirani, R.: Discriminant analysis by Gaussian mixtures. Journal of the Royal Statistical Society. Series B **58**(1), 155–176 (Jul 1996)
8. Huang, P., Yuan, Y., Lan, Z., Jiang, L., Hauptmann, A.G.: Video representation learning and latent concept mining for large-scale multi-label video classification. arXiv:1707.01408 (2017)
9. Inoue, H.: Data augmentation by pairing samples for images classification. arXiv:1801.02929 (2018)
10. Kohonen, T.: Learning vector quantization. In: Springer (ed.) Self-Organizing Maps, chap. 6. Springer Series in Information Sciences, Berlin, Heidelberg (1995)
11. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep., Department of Computer Science, University of Toronto (2009)
12. Le Cun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (Nov 1998)
13. Na, S., Yu, Y., Lee, S., Kim, J., Kim, G.: Encoding video and label priors for multi-label video classification on YouTube-8M dataset. In: CVPR Workshops (2017)
14. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NIPS Workshops. pp. 2999–3007. Venice, Italy, (Oct 2017)
15. Sarafianos, N., Xu, X., Kakadiaris, I.A.: Deep imbalanced attribute classification using visual attention aggregation. In: ECCV. Munich, Germany (Sep 2018)
16. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR. San Diego, CA, USA (May 2015)
17. Skalic, M., Austin, D.: Building a size constrained predictive model for video classiffication. In: CVPR Workshops (2017)
18. Skalic, M., Pekalski, M., Pan, X.E.: Deep learning methods for efficient large scale video labeling. In: CVPR Workshops (2017)
19. Soudry, D., Hoffer, E., Nacson, M.S., Gunasekar, S., Srebro, N.: The implicit bias of gradient descent on separable data. JMLR **19**(70), 1–57 (Jan 2018)
20. You, D., Hamsici, O.C., Martinez, A.M.: Kernel optimization in discriminant analysis. IEEE Trans. Pattern Anal. Mach. Intell. **33**(3), 631–638 (Mar 2011)
21. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: BMVC. York, UK, (Sep 2016)