ELoPE: Fine-Grained Visual Classification with Efficient Localization, Pooling and Embedding

Harald Hanselmann[1,2] and Hermann Ney[1,2]
[1]Human Language Technology and Pattern Recognition Group
Computer Science Department, RWTH Aachen University, 52062 Aachen, Germany
[2]AppTek GmbH, 52062 Aachen, Germany
{hanselmann, ney}@cs.rwth-aachen.de

Abstract

The task of fine-grained visual classification (FGVC) deals with classification problems that display a small inter-class variance such as distinguishing between different bird species or car models. State-of-the-art approaches typically tackle this problem by integrating an elaborate attention mechanism or (part-) localization method into a standard convolutional neural network (CNN). Also in this work the aim is to enhance the performance of a backbone CNN such as ResNet by including three efficient and lightweight components specifically designed for FGVC. This is achieved by using global k-max pooling, a discriminative embedding layer trained by optimizing class means and an efficient localization module that estimates bounding boxes using only class labels for training. The resulting model achieves state-of-the-art recognition accuracies on multiple FGVC benchmark datasets.

1. Introduction

Fine-grained visual classification (FGVC) refers to classification tasks where the differences between the different categories are very subtle. Examples of such tasks are the classification of bird species or differentiating between different car models. The general appearance of the categories is very similar (*e.g.* all birds have two wings and a beak, cars typically have four wheels) and as result the inter-class variation is small. On the other hand, the intra-class variation can be quite high (*e.g.* due to different poses). This makes FGVC a very challenging problem that receives a lot of attention in the research community. State-of-the-art approaches typically involve a backbone CNN such as ResNet [11] or VGG [25] that is extended by a method that localizes and attends to specific discriminative regions. These methods can become quite complex and sometimes require multiple passes through the backbone CNN.

In this work we aim to improve the performance of a given backbone CNN with little increase in complexity and requiring just a single pass through the backbone network during testing. Specifically, we propose to the following three steps:

- Global k-max pooling: For FGVC-models, the final convolutional layer often still has a spatial resolution of $I \times J$ (e.g. for a ResNet-50 with 448×448 input images the resolution is 14×14). A single feature vector describing the image can then be obtained by using global average or global max pooling. However, to approximate part-based recognition, we propose to use global k-max pooling [14], where the average over the k maximal activations is computed.

- Embedding layer: In a typical setup for face verification tasks, the test subjects (*i.e.* classes) are not known during training, which means a standard softmax classifier can not be trained. CNNs are therefore often used to train a discriminative embedding space in which face images can be compared efficiently and accurately. The embeddings are learned using specifically designed loss functions such as center loss [30], triplet loss [23] or DFF [10]. We insert such an embedding layer trained with a loss function similar to [10] into the backbone CNN as penultimate layer. We show that this greatly improves the performance of the softmax classifier.

- Localization module: Using bounding boxes to crop the input images typically improves the performance of the classification model. In order to avoid having to rely on human bounding box annotations we train an efficient bounding box detector that can be applied before the image is processed by the backbone CNN. This localization module is lightweight and trained using only the class labels. Bounding box annotations are not needed.

We evaluate our model on three popular FGVC datasets from different domains. The first dataset is CUB200-2011 [28] where the task is the classification of bird species. The second dataset is Stanford cars [16] where different car models are classified and the third is FGVC-Aircraft [21] for the classification of different aircraft models. We obtain very competitive results on all three datasets and to the best of our knowledge state-of-the-art results for the latter two.

1.1. Related work

As mentioned in the introduction FGVC has received a lot of attention in the research community. As a result, many different approaches have been proposed. Especially using some form of visual attention has been very popular lately [37, 36, 34, 7, 32, 18, 27]. The work presented in [27] is of particular relevance since here also an embedding loss is used, but with the aim to guide the defined attention mechanism.

Spatial transformations that extract the discriminative parts of the input can also be seen as a form of attention. For example, the well known spatial transformer introduced in [13] is capable of learning global transformations (*e.g*. affine transformations), but is known to be difficult to train and usually needs a second large network to estimate the transformation parameters. In [33] a module to learn pixel-wise translations is proposed. However, this module is applied very late in the network, possibly due to being reliant on high-level features. As a result, only the last layers can profit from the localized input. In [24] an ensemble of networks is learned sequentially, where each network is trained based on a spatial transformation derived from the previous network. This means that each input image needs to be passed through multiple networks. Our localization module fits into the category of spatial transformations (limited to scale and translation), but it is very lightweight and easy to train while still being able to significantly boost the recognition performance.

The work presented in [19] proposes to train a Gaussian mixture model based on part proposals provided by selective search. However, this requires a looped training procedure with the EM-algorithm.

Another popular approach is based on bilinear models [20] which can lead to issues with efficiency due to very high dimensional features and multi-stream architectures. Also other second-order pooling methods such as the work in [17] can result in very high dimensional features.

Other approaches include learning global and patch features in an asymmetric multi-stream architecture [29], learning a complex sequence of data augmentation steps from the data [3], deep layer aggregation [35] or the training of very large networks (*e.g*. 557 million parameters) [12]. It is also possible to boost the performance by obtaining more training data [4, 15].

2. Overview

The goal of this work is to improve the performance of a given backbone CNN (*e.g*. a ResNet [11]) with lightweight components that do not require multiple passes through the backbone CNN or significantly higher runtime or memory usage during testing. We achieve this goal by adding three components, a localization module, global k-max pooling [14] and an embedding layer. An overview of the resulting model in the testing stage is given in Figure 1. The input image that needs to be classified is forwarded through the localization module which estimates the bounding box of the object in the image and returns a cropped image. This cropped image is then forwarded through the backbone CNN that contains global k-max pooling and the embedding layer at the later stages. The classification result is then given by a softmax classification layer.

In the training stage the model is trained jointly with a standard cross-entropy L_{CE} applied at the classification layer and a specific loss function L_e that is applied at the embeddings layer:

$$L = L_{CE} + \lambda L_e \tag{1}$$

The localization module is trained separate training step (c.f. Section 5).

3. Global K-Max Pooling

Often global max pooling (GMP) or global average pooling (GAP) is used between the last convolutional layer and the classification layer of a CNN. These pooling operations allow to break down the spatial dimension of the final convolutional layer and obtain a single vector describing the image. For FGVC it has been shown that part-based approaches (*e.g*. [37]) can boost the classification performance. For this reason we propose to use global k-max pooling [14] (GKMP). This two-step pooling procedure first applies k-max pooling at the last convolutional layer which is followed by an averaging operation over the K selected maximal values in each feature map. This way the network can learn features that activate at the K most important parts of the image (during back-propagation the error gets propagated through the K most important parts instead of just one as with GMP or all of them as with GAP). This could be seen as a very simple form of attention.

The global k-max pooling layer can be defined as follows. Given an input image x, let $y \in \mathbb{R}^{D \times I \times J}$ be the output of the last convolutional layer of a CNN, where y has a spatial resolution of $I \times J$ (in this work typically 14×14) and contains D feature maps. Further, given a specific $d \in \{1, ..., D\}$ the sorted vector S_d contains the

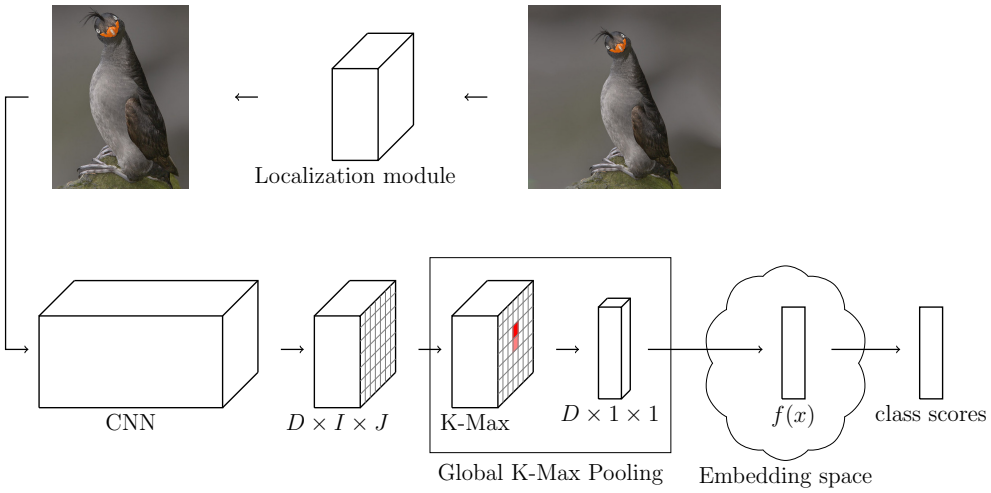

Figure 1. Overview of the proposed model including a lightweight localization module, global k-max pooling and an embedding layer.

values at y_d sorted in descending order:

$$S_d = sorted\_desc(\{y_{d,i,j} | i \in \{1, ..., I\}, \\ j \in \{1, ..., J\}\}) \quad (2)$$

The global k-max pooling operation for a specified K is then is then defined as:

$$GKMP(y)_d = \frac{1}{K} \sum_{k=1}^{K} S_{d,k} \quad (3)$$

This definition corresponds to the pooling operation used in [6], except in [6] also the K minimal activations are included. However, preliminary experiments suggest that including the minimal activations does not help the recognition performance in our case. Therefore we use global k-max pooling as defined in Equation (3).

There are two special cases of GKMP that are worth mentioning. If we select $K = 1$ then Equation (3) results in the standard GMP, while a choice of $K = I \cdot J$ leads to GAP. In this work we chose $K = 4$ in all experiments.

Note that GKMP is only used after the last convolutional layer with the aim to summarize the image into one vector. It does not replace other pooling operations within the backbone network.

3.1. Global k-max pooling with weighted averaging

GKMP can be extended to a global weighted k-max pooling variant (GWKMP) by including weights in the averaging operation in Equation (3) similar to [31]:

$$GKMP(y)_d = \frac{1}{K} \sum_{k=1}^{K} w_k \cdot S_{d,k} \quad (4)$$

This formulation extends the network with only K parameters that regulate the contribution of the K maximal activations to the averaging operation in each feature map.

4. Embedding layer

The embedding layer is inserted between GKMP or GWKMP and the classification layer. The idea is to map the images into a discriminative embedding space, where the distances between images of the same class are small, while the distances between images of different classes are large. This concept is known from face verification [23, 30, 10] and metric learning [22]. Different from those two tasks, we do not compare images directly in the embedding space, but use it as an intermediate layer (more specifically as penultimate layer). The classification is done by a standard classification layer trained on the embedding space with cross-entropy. We argue that one of the advantages of this approach is that it can help mitigate the issue of limited training data that we often see in FGVC tasks (see Table 1).

The embedding space is trained using a specific loss function L_e applied directly to output of this intermediate layer. In this work we use a formulation based on optimizing class means [30, 10] since this is easy to integrate into the training and does not require any specific batch construction schemes (unlike tuple-based losses such as the triplet loss [23] or the N-Pair loss [26]). For each class c a feature vector is computed (and updated online during training) that describes the class mean μ_c within the embedding space. The goal of the loss function L_e is to minimize the distance of each image within a batch to its respective class mean, while maximizing the distance between means of different classes.

The loss function is composed of two parts:

$$L_e = L_w + L_b \tag{5}$$

The first part L_w is the within-class (intra-class) loss that minimizes the distances of the images to their class means, while the second part L_b is the between-class (inter-class) loss that maximizes the distances between class means.

4.1. Within-class loss

We use the same formulation for the within-class loss as in [10] (which is derived from the center-loss proposed in [30]) including an additional $l2$-normalization. Given classes $c \in \{1, ..., C\}$ and a batch of images x_n with $n \in \{1, ..., N\}$ let $f(x_n)$ be the normalized output of the embedding layer. Assuming we are currently in training iteration t the first step is to update the class means using the data points in the current batch and the class means from the previous iteration $t-1$

$$\mu_c^t = \mu_c^{t-1} - \alpha \Delta \mu_c^{t-1} \tag{6}$$

where the hyper-parameter α can be considered the learning rate of the class means. The term $\Delta \mu_c^{t-1}$ is defined as

$$\Delta \mu_c^{t-1} = \frac{\sum_{n=1}^{N} \delta(c_n, c)(\mu_{c_n}^{t-1} - f(x_n))}{1 + \sum_{n=1}^{N} \delta(c_n, c)} \tag{7}$$

and $\delta(c_n, c)$ is the Kronecker-function:

$$\delta(c_n, c) = \begin{cases} 1 & c = c_n \\ 0 & c \neq c_n \end{cases} \tag{8}$$

Note that this update creates a functional dependence between the class means and their corresponding images within the batch which has to be considered during back-propagation [10].

The within-class loss function is then defined as

$$L_w = \frac{1}{2N} \sum_{n=1}^{N} \| f(x_n) - \mu_{c_n}^t \|_2^2 \tag{9}$$

4.2. Between-class loss

The between-class loss maximizes the distances between the updated class means:

$$L_b = \frac{\gamma}{4|P|} \sum_{(k,c) \in P} \max\left(m - \|\mu_k^t - \mu_c^t\|_2^2, 0\right)^2 \tag{10}$$

The terms μ_k^t and μ_c^t are the new class means updated with the features $f(x_n)$ from the current batch as computed in Equation (6). The margin m defines a threshold for the distances to be penalized and γ controls the contribution of the between-class part to the embedding loss L_e. The set P with cardinality $|P|$ contains all class-pairs in the current batch.

To see why squaring the maximum in Equation (10) is important we have a look at the gradient with respect to $f(x_n)$:

$$\frac{\partial L_b}{\partial f(x_n)} = \sum_{(k,c) \in P} \begin{cases} 0 & m \leq \|\mu_k^t - \mu_c^t\|_2^2 \\ grad(m, \mu_k^t, \mu_c^t) & \text{otherwise} \end{cases} \tag{11}$$

with

$$grad(m, \mu_k^t, \mu_c^t) = \underbrace{(m - \|\mu_k^t - \mu_c^t\|_2^2)}_{d(m, \mu_k^t, \mu_c^t)} \frac{\partial}{\partial f(x_n)} \left(\frac{\gamma}{2|P|} \|\mu_k^t - \mu_c^t\|_2^2 \right) \tag{12}$$

The squared maximization leads to the appearance of the term $d(m, \mu_k^t, \mu_c^t)$ in the gradient. As a result the gradient gets larger if the distance between two class means gets smaller which encourages the model to focus more on improving the distance of class-pairs with very close means. This should help to reduce classification errors due to the confusion of images from class-pairs with close class means.

While the formulation of the between-class loss defined above is close to [10], the appearance of $d(m, \mu_k^t, \mu_c^t)$ in the gradient is a key difference, since in [10] the maximization in the between-class loss is not squared. In addition there are two more differences. First, in this work the centers are based on $l2$-normalized features, which makes the choice for the margin m easier, since distances between $l2$-normalized vectors are restricted by a certain range. The second difference is that in [10] the set P contains a sampled subset of all class-pairs in the current batch. Since we work with much smaller batch sizes (see Section 6) compared to [10], we use all pairs within a batch.

The embedding layer as defined above is also closely related to the attention regulation proposed in OSME-MAMC [27]. The latter uses a metric learning loss to learn correlations between the output of different attention branches. If the multiple attention branches were replaced by a single fully connected layer then this method would be equivalent to training an embedding layer in our framework with the N-Pair loss [26].

5. Localization module

It can be observed that cropping the images based on bounding boxes of the objects that need to be recognized

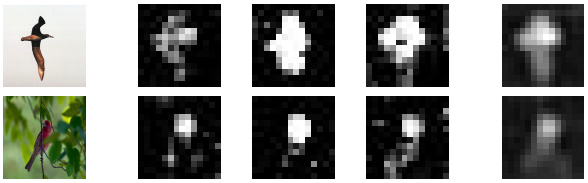

Figure 2. Min-max normalized activations of the last convolutional layer of a trained model. The three images in the middle are the activations in a specific feature map, while the last image shows the mean over all feature maps.

can improve the classification performance. Since we do not want to use any annotations apart from the class labels in training (and of course no annotations at all during testing) we need a way to obtain these localizations without any additional annotations if we want to profit from the increased performance the localizations can provide. We design such a localization module based on the following observations.

The final convolutional layer of a trained classification model typically contains higher activations in positions corresponding the discriminative areas of the image (see Figure 2). By computing the mean over all feature maps a quite accurate heat map can be computed for the object in the image. This heat map can the be used to find the boundaries of the object within the final layer. A similar observation has been made in [38], but in contrast to [38] we do not consider class-specific heat maps due to the small inter-class variations in FGVC.

The downside with this approach is that we obtain these bounding boxes only at the final layer while the full image needs to be forwarded through all previous layers. One option would be to apply ROI-Pooling after the final layer based on the estimated bounding box. However, this would mean that only the fully connected layers following the last convolutional layer would profit from focusing on the discriminative area of the image. All other layers still have to operate on the full image and we can not fully exploit the potential from using the bounding boxes (see Figure 5). An alternative would be to pass the image through the full backbone network, extract the bounding boxes and then train a second, more precise model based on the bounding boxes (similar to [24]). However, with this approach the image would have to be passed through a full network twice, one pass to obtain the bounding box and a second pass for classification. This is not very efficient. To avoid such a multi-pass procedure we propose to train a very lightweight localization module that predicts the bounding boxes and is integrated into the backbone network such that an image can be processed in one pass.

The architecture of the localization module is equivalent to the first few layers of a ResNet-50 (initialized from the trained classification model) until the end of the first residual block including an initial down-sampling layer that resizes the input to a spatial resolution of 64×64. The module has only 220000 parameters and can be added to the classification model without taking up much runtime or memory. The output is of the size $1 \times I \times J$, the same as the mean of the last convolutional layer of the trained classification model. The localization module is trained by feeding an input image through a trained classification model and the localization module. The outputs are compared using the smooth L1 loss [9] (see Figure 3). During back-propagation the weights of the trained classification model are fixed and only the localization module is trained. This way the localization module learns to directly predict the heat maps.

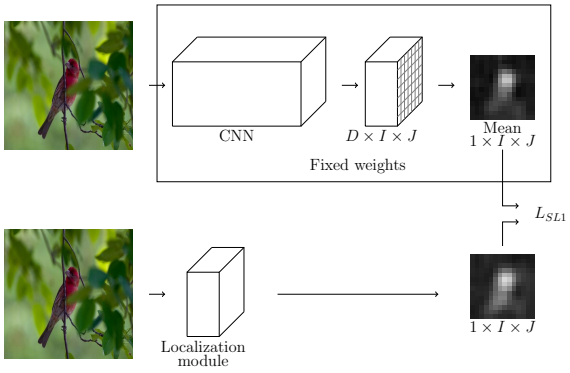

Figure 3. Training procedure for the localization module: The localization module learns to directly predict the heat maps generated by the backbone CNN.

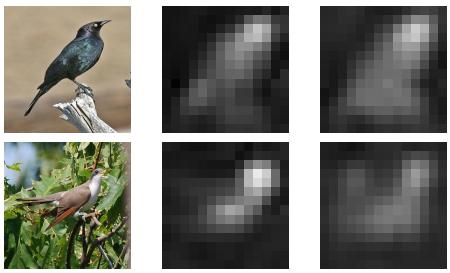

Figure 4. True mean (middle) of the trained classification model compared to estimated mean (right).

Examples of the estimated heat maps are given in Figure 4. The localization module is able to predict the heat maps very accurately, even though less focused on a specific part of the bird (especially in the second and third row).

To obtain the final bounding box we process the heat map using min-max normalization and binarization based on a given threshold τ. The bounding box is then the smallest rectangle containing all pixels where the heat map has a value that is greater than τ.

Name	#Train images	#Test images	#Classes
CUB200-2011 [28]	5994	5794	200
Stanford cars [16]	8144	8041	196
FGVC-Aircraft [21]	6667	3333	100

Table 1. Datasets used for the evaluation.

6. Experimental evaluation

We evaluate our proposed approach on three popular datasets for fine-grained classification, CUB200-2011 [28] for bird-species classification, Stanford cars [16] for the classification of car models and FGVC-Aircraft [21] for the classification of airplane models. The statistics of the three datasets are given in Table 1. On all datasets we use only the class label annotations. We do not use any additional annotations such as bounding boxes or part annotations.

As backbone CNN we select ResNet [11] where we try the two variants ResNet-50 and ResNet-101 (the analysis in Section 6.2 and 6.3 is done using ResNet-50 as backbone CNN). The models are pre-trained on the ImageNet [5] from which we remove all images that overlap with the test sets of the three FGVC tasks used for this evaluation.

Since our added components are all lightweight and do not occupy much memory we are able to train both ResNet variants on a single NVIDIA GeForce® GTX 1080 Ti GPU with 11GB memory with a batch-size of 14 and input images with the spatial resolution 448×448. We use this batch-size and input resolution for all experiments. The models are trained with standard back-propagation for 90 epochs with momentum of 0.9 and weight decay of 0.001. The starting learning rate is 0.003 which is reduced by a factor of 10 after 30 epochs. The weighted average pooling (see Section 3.1) is applied as finetuning step for 30 more epochs. The approach is implemented using the Torch7 framework [2] and the code will be made available[1].

There is a number of hyper-parameters to set for our approach. We perform a very limited search for the hyper-parameters on CUB200-2011 using a validation set separated from the training images. The parameters are quite robust and we can use the same set of hyper-parameters in all other experiments. The threshold τ for the localization module is the only exception, where we use a slightly smaller value on the Stanford cars and the FGVC-Aircraft dataset. The exact values for the hyper-parameters are given in Table 2.

In Section 6.4 and 6.5 we compare to state-of-the-art approaches using a similar experimental setting, specifically with the same training data (ImageNet and the training set of the FGVC task at hand) and no bounding box or part annotations. Note that for some datasets better results can be achieved by acquiring large amounts of additional training data [4, 15].

[1] https://github.com/rwth-i6/fgvc

Hyper-parameter	Reference	Value
α	Equation (6)	0.5
λ	Equation (1)	2.0
γ	Equation (10)	16.0
m	Equation (10)	0.75
K	Equation (3)	4
τ	Section (5)	0.3, 0.2

Table 2. Hyper-parameters used in the experiments. The threshold τ is 0.3 on CUB200-2011 and 0.2 on the other two datasets.

Method	Runtime [ms]	Parameters
Baseline	96.8	24M
Ours	101.3	25M

Table 3. Computational efficiency comparison between baseline ResNet-50 (input $14 \times 3 \times 448 \times 448$) and our approach.

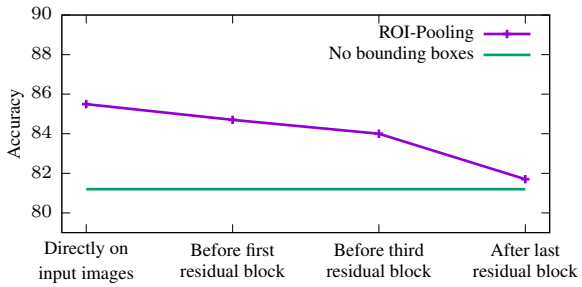

Figure 5. Accuracies for ROI-Pooling with ground-truth bounding boxes applied at different depths of a CNN on CUB200-2011 (ResNet-50 with GAP and no embedding learning).

6.1. Computational efficiency

A comparison of the computational efficiency between a baseline ResNet-50 and our approach with ResNet-50 as backbone is given in Table 3. With a typical input resolution the additional complexity in terms of runtime caused by our approach is small. The numbers were measured for one forward pass on a NVIDIA GeForce® GTX 1080 Ti GPU using an input of dimension $14 \times 3 \times 448 \times 448$. There is also no large increase in parameters highlighting the efficiency of the proposed approach.

6.2. Localization module

In Figure 5 we analyze the effect on the recognition performance if the image or feature maps are cropped based on ground-truth bounding boxes (ROI-Pooling). We can observe that the earlier the ROI-Pooling is applied the better is the recognition accuracy, confirming our expectation in Section 5.

Method	Accuracy[%]
GoogleNet-GAP [38]	41.0
ResNet-50 mean feature map	58.6
Localization module	68.9

Table 4. Accuracy of localization if intersection over union (IoU) is at least 0.5.

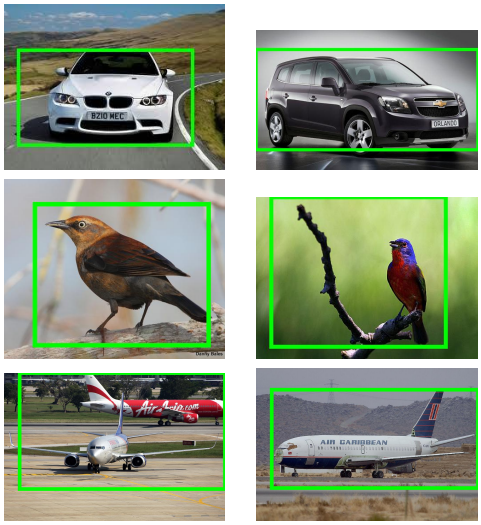

Figure 6. Examples of bounding boxes detected by the localization module.

The accuracy of the bounding boxes can be calculated by counting a bounding box as correct if the intersection over union (IoU) with the ground truth bounding box is at least 0.5 [38]. We can observe in Table 4 that using the mean over baseline ResNet-50 feature maps of the last convolutional layer already achieves a better accuracy then what is reported in [38]. However, on top of being much more efficient the localization module is also even more accurate. We argue that this is due to the slightly more general heatmaps generated by the localization module as a result of the approximation (see Figure 4).

In Figure 6 we show qualitative results of the bounding boxes estimated for validation images by the localization module. We can observe that the localization module is able to find quite accurate bounding boxes. In some cases the localization module over-estimates the bounding box due to other objects in the image such as a branch or a distracting background. Another interesting reason for over-estimated bounding boxes can be multiple instances of the given FGVC domain such as two different airplane models in the same image. However, in each of these cases the classifier still has a good chance of finding the correct class, since the objects are still present in the cropped image.

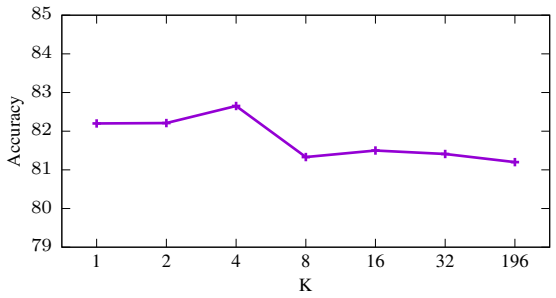

Figure 7. Accuracies for different values of K in K-max pooling on CUB200-2011.

6.3. Ablation study

Table 5 shows how the components contribute to the recognition performance. The two baseline results with GAP and GMP include a fully connected layer with dimension 512 before the classification layer. We can observe in Table 5 that adding an embedding loss L_e and adding the localization module leads to the largest boost in performance at about 2% absolute (86.9%, using ground-truth bounding boxes yields 87.5%). The addition of global k-max pooling, weighted average finetuning and the full embedding loss compared to only the within-class part leads to an improvement of about 0.5% each.

Pooling	L_e	Loc. module	Accuracy[%]
GAP			81.2
GMP			82.2
GKMP			82.7
GKMP		✓	84.2
GKMP	Center-loss [30]		84.1
GKMP	DFF [10]		84.6
GKMP	L_w		84.4
GKMP	$L_w + L_b$		84.9
GKMP	$L_w + L_b$	✓	86.9
GWKMP	$L_w + L_b$	✓	87.4

Table 5. Results on CUB200-2011 with a ResNet-50. The notation $L_e = L_w$ means the embedding layer is trained only with the within-class part of the embedding loss, while $L_e = L_w + L_b$ means the embedding layer is trained with the full embedding loss. For comparison we include results obtained by using the loss functions defined in [30] and [10] in our framework.

The effect of the value selected for K in GKMP is illustrated in Figure 7. The special case $K = 1$ is equivalent to GMP and $K = 196$ is equivalent to GAP (the spatial dimension of the last convolutional layer is 14×14). We can observe that a value around 4 seems to be optimal. The improvement compared to GMP is only 0.5% absolute, but using a value larger than 1 also enables us to apply the

weighted averaging, which gives another performance boost (see Table 5).

6.4. CUB200-2011

Method	Backbone CNN	Accuracy[%]
STN [13]	BN-Inception	84.1
MA-CNN [37]	VGG-19	86.5
GMM [19]	VGG-19	86.3
Spatial RNN [32]	M-Net/D-Net	89.7
Stacked LSTM [8]	GoogleNet	**90.4**
DLA [35]	DLA-102	85.1
FAL [33]	ResNet-50	84.2
DT-RAM [18]	ResNet-50	86.0
ISE [24]	ResNet-50	87.2
DFL-CNN [29]	ResNet-50	87.4
NTS-Net [34]	ResNet-50	87.5
DCL [1]	ResNet-50	87.8
OSME-MAMC [27]	ResNet-101	86.5
iSQRT-COV [17]	ResNet-101	88.7
Ours	ResNet-50	87.4
Ours	ResNet-101	88.5

Table 6. Comparison of our approach with other state-of-the-art methods on CUB200-2011.

In Table 6 we compare our approach to other state-of-the-art methods. In addition to the results reported in Table 5 we also evaluate our method with ResNet-101 as backbone network. This includes all components (GKMP with weighted average pooling, embedding layer with full embedding loss, localization module). Our best result is with 88.5% very competitive, even though a little less accurate then the state-of-the-art results that are reported in [32] and [8]. However, these involve recurrent neural networks which can be computationally expensive.

6.5. Stanford cars and FGVC-Aircraft

The results on the Stanford cars and the FGVC-Aircraft dataset are given in Table 7 and 8, respectively. Here we run the experiments with ResNet-50 and ResNet-101 with all components included. As mentioned earlier, we use the same hyper-parameters as for CUB200-2011 (apart from the threshold τ). Again, our approach achieves very competitive state-of-the-art results for both ResNet variants.

7. Conclusion

In this work we presented an efficient method to improve the classification performance of a backbone CNN for fine-grained visual classification. Specifically, we propose a lightweight localization module that relies only on class label annotations during training. We showed that the localization module can find reliable bounding boxes and significantly boost the recognition performance. Additionally we propose to use global k-max pooling [14] to obtain a global vector describing the image. This approximates part-based modeling and can further be improved by learning weights to regulate the contribution of the maximal values in each feature map. Finally, we project the image descriptor into a discriminative embedding space from which the classification layer makes the classification. As an intermediate layer of the full classification network the embedding space is trained jointly with the full network and a specific loss function that optimizes class means. We evaluate our approach on three popular FGVC tasks and achieve competitive results on all three. Especially on Stanford cars and FGVC-Aircraft we can report state-of-the-art classification accuracies.

Method	Backbone CNN	Accuracy[%]
MA-CNN [37]	VGG-19	92.8
GMM [19]	VGG-19	93.5
DFL-CNN [29]	VGG-16	93.8
Spatial RNN [32]	M-Net/D-Net	93.4
DLA [35]	DLA-X-60-C	94.1
DT-RAM [18]	ResNet-50	93.1
ISE [24]	ResNet-50	94.1
NTS-Net [34]	ResNet-50	93.9
DCL [1]	ResNet-50	94.5
GPipe [12]	AmoebaNet-B	94.8
AutoAugm [3]	Inception-v4	94.8
OSME-MAMC [27]	ResNet-101	93.0
iSQRT-COV [17]	ResNet-101	93.3
Ours	ResNet-50	94.5
Ours	ResNet-101	**95.0**

Table 7. Comparison of our approach with other state-of-the-art methods on Stanford cars.

Method	Backbone CNN	Accuracy[%]
MA-CNN [37]	VGG-19	89.9
GMM [19]	VGG-19	90.5
DFL-CNN [29]	VGG-16	92.0
Spatial RNN [32]	M-Net/D-Net	88.4
DLA [35]	DLA-X-60	92.9
ISE [24]	ResNet-50	90.9
NTS-Net [34]	ResNet-50	91.4
DCL [1]	ResNet-50	93.0
GPipe [12]	AmoebaNet-B	92.9
AutoAugm [3]	Inception-v4	92.7
iSQRT-COV [17]	ResNet-101	91.4
Ours	ResNet-50	93.4
Ours	ResNet-101	**93.5**

Table 8. Comparison of our approach with other state-of-the-art methods on FGVC-Aircraft.

References

[1] Y. Chen, Y. Bai, W. Zhang, and T. Mei. Destruction and construction learning for fine-grained image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2019.

[2] R. Collobert, K. Kavukcuoglu, C. Farabet, et al. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS workshop*, volume 5, page 10. Granada, 2011.

[3] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2019.

[4] Y. Cui, Y. Song, C. Sun, A. Howard, and S. Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4109–4118, June 2018.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.

[6] T. Durand, N. Thome, and M. Cord. Weldon: Weakly supervised learning of deep convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4743–4752, June 2016.

[7] J. Fu, H. Zheng, and T. Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4438–4446, July 2017.

[8] W. Ge, X. Lin, and Y. Yu. Weakly supervised complementary parts models for fine-grained image classification from the bottom up. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2019.

[9] R. Girshick. Fast r-cnn. In *IEEE International Conference on Computer Vision*, pages 1440–1448, December 2015.

[10] H. Hanselmann, S. Yan, and H. Ney. Deep fisher faces. In *British Machine Vision Conference*, September 2017.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE conference on Computer Vision and Pattern Recognition*, pages 770–778, June 2016.

[12] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, and Z. Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in Neural Information Processing Systems*, pages 103–112, December 2019.

[13] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, December 2015.

[14] P. Koniusz, F. Yan, and K. Mikolajczyk. Comparison of mid-level feature coding approaches and pooling strategies in visual concept detection. *Computer vision and image understanding*, 117(5):479–492, 2013.

[15] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei. The unreasonable effectiveness of noisy data for fine-grained recognition. In *European Conference on Computer Vision*, pages 301–320, October 2016.

[16] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *IEEE International Conference on Computer Vision Workshops*, pages 554–561, December 2013.

[17] P. Li, J. Xie, Q. Wang, and Z. Gao. Towards faster training of global covariance pooling networks by iterative matrix square root normalization. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2018.

[18] Z. Li, Y. Yang, X. Liu, S. Wen, and W. Xu. Dynamic computational time for visual attention. *IEEE International Conference on Computer Vision Workshops*, pages 1199–1209, October 2017.

[19] J. Liang, J. Guo, X. Liu, and S. Lao. Fine-grained image classification with gaussian mixture layer. *IEEE Access*, 6:53356–53367, 2018.

[20] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear cnn models for fine-grained visual recognition. In *IEEE International Conference on Computer Vision*, pages 1449–1457, December 2015.

[21] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[22] O. Rippel, M. Paluri, P. Dollar, and L. Bourdev. Metric learning with adaptive density discrimination. *International Conference on Learning Representations*, May 2016.

[23] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, June 2015.

[24] A. Simonelli, F. De Natale, S. Messelodi, and S. R. Bulo. Increasingly specialized ensemble of convolutional neural networks for fine-grained recognition. In *IEEE International Conference on Image Processing*, pages 594–598, October 2018.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, May 2015.

[26] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1857–1865, December 2016.

[27] M. Sun, Y. Yuan, F. Zhou, and E. Ding. Multi-attention multi-class constraint for fine-grained image recognition. In *European Conference on Computer Vision*, pages 805–821, September 2018.

[28] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[29] Y. Wang, V. I. Morariu, and L. S. Davis. Learning a discriminative filter bank within a cnn for fine-grained recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4148–4157, June 2018.

[30] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515, October 2016.

[31] C. Weng, H. Wang, and J. Yuan. Learning weighted geometric pooling for image classification. In *IEEE International Conference on Image Processing*, pages 3805–3809, September 2013.

[32] L. Wu, Y. Wang, X. Li, and J. Gao. Deep attention-based spatially recursive networks for fine-grained visual recognition. *IEEE Transactions on Cybernetics*, 49(5):1791–1802, 2018.

[33] Q. Xu, Y. Sun, Y. Li, and S. Wang. Attend and align: Improving deep representations with feature alignment layer for person retrieval. In *International Conference on Pattern Recognition*, pages 2148–2153, August 2018.

[34] Z. Yang, T. Luo, D. Wang, Z. Hu, J. Gao, and L. Wang. Learning to navigate for fine-grained classification. In *European Conference on Computer Vision*, September 2018.

[35] F. Yu, D. Wang, E. Shelhamer, and T. Darrell. Deep layer aggregation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2403–2412, June 2018.

[36] B. Zhao, X. Wu, J. Feng, Q. Peng, and S. Yan. Diversified visual attention networks for fine-grained object classification. *IEEE Transactions on Multimedia*, 19(6):1245–1256, 2017.

[37] H. Zheng, J. Fu, T. Mei, and J. Luo. Learning multi-attention convolutional neural network for fine-grained image recognition. In *IEEE International Conference on Computer Vision*, pages 5209–5217, October 2017.

[38] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, June 2016.