Take 5: Interpretable Image Classification with a Handful of Features

Thomas Norrenbrock  Marco Rudolph  Bodo Rosenhahn
Institute for Information Processing (tnt)
L3S - Leibniz Universität Hannover, Germany
{norrenbr, rudolph, rosenhahn}@tnt.uni-hannover.de

Abstract

Deep Neural Networks use thousands of mostly incomprehensible features to identify a single class, a decision no human can follow. We propose an interpretable sparse and low dimensional final decision layer in a deep neural network with measurable aspects of interpretability and demonstrate it on fine-grained image classification. We argue that a human can only understand the decision of a machine learning model, if the features are interpretable and only very few of them are used for a single decision. For that matter, the final layer has to be sparse and - to make interpreting the features feasible - low dimensional. We call a model with a Sparse Low-Dimensional Decision “SLDD-Model”. We show that a SLDD-Model is easier to interpret locally and globally than a dense high-dimensional decision layer while being able to maintain competitive accuracy. Additionally, we propose a loss function that improves a model’s feature diversity and accuracy. Our more interpretable SLDD-Model only uses 5 out of just 50 features per class, while maintaining 97% to 100% of the accuracy on four common benchmark datasets compared to the baseline model with 2048 features.

Figure 1: Local explanation by our SLDD-Model: The two features used for the predicted class, emerged without additional supervision, are aligned with human interpretable attributes and localized (described in App. D) adequately.

1 Introduction

Understanding the decision of a deep learning model is becoming more and more important. Especially for safety-critical applications such as the medical domain or autonomous driving, it is often either legally (Bibal et al., 2021) or by the practitioners required to be able to trust the decision and evaluate its reasoning (Molnar, 2020). Due to the high dimensionality of images, most previous work on interpretable models for computer vision combines the deep features computed by a deep neural network with a method that is considered interpretable, such as a prototype based decision
tree (Nauta et al., 2021). While approaches for measuring the interpretability without humans exist for conventional machine learning algorithms (Islam et al., 2020), they are missing for methods including deep neural networks. In this work, we propose a novel sparse and low-dimensional SLDD-Model which offers measurable aspects of interpretability. The key aspect is a heavily reduced number of features, out of which only very few are considered per class. Humans can only consider $7 \pm 2$ aspects at once (Miller, 1956) and could therefore follow a decision that uses that many features. To be intelligible for all humans, we aim for an average of 5 features per class.

Having a reduced number of features makes it feasible to investigate every single feature and understand its meaning: We are able to align several of the learned features with human concepts post-hoc. The combination of reduced features and sparsity therefore increases both global and local interpretability. How does the model behave? and local interpretability Why did the model make this decision?, demonstrated in Figure 1.

Our proposed method generates the SLDD-Model by utilizing glm-saga (Wong et al., 2021) to compute a sparse linear classifier for selected features, which we then finetune to the sparse structure. We apply feature selection instead of a transformation to reduce the computational load and preserve the original semantics of the features, which can improve interpretability (Tao et al., 2015), especially if a more interpretable model like B-cos Networks (Böhle et al., 2022) is used. Additionally, we propose a novel loss function for more diverse features, which is especially relevant when one class depends on very few features, since using more redundant features limits the total information available for the decision.

Our main contributions are as follows:

- We present a pipeline that ensures a model with increased global and local interpretability which identifies a single class with just few, e.g., 5, features of its low-dimensional representation. We call the resulting model SLDD-Model.
- Our novel feature diversity loss ensures diverse features. This increases the accuracy for the extremely sparse case.
- We demonstrate the competitive performance of our proposed method on four common benchmark datasets in the domain of fine-grained image classification as well as ImageNet-1K (Russakovsky et al., 2015), and show that several learned features for algorithmic decision-making can be directly connected to attributes humans use.

2 Related Work

2.1 Fine-Grained Image Classification

Fine-grained image classification describes the problem of differentiating similar classes from one another. It is more challenging compared to conventional image recognition tasks (Lin et al., 2015) since the differences between classes are much smaller. To tackle this difficulty, several adaptions to the common image classification approach have been applied. They usually involve learning more discriminative features by adding a term to the loss function (Chang et al., 2020; Liang et al., 2020; Zheng et al., 2020), introducing hierarchy to the architecture (Chou et al., 2022) or using expensive expert knowledge (Chen et al., 2018; Chang et al., 2021). Chang et al. (2020) divide the features into groups, s.t. every group is assigned to exactly one class. While training, an additional loss increases the activations of features for samples of their assigned class and reduces the overlap of feature maps in each group. Liang et al. (2020) tried to create class-specific filters by inducing sparsity in the features. Both (Chang et al., 2020) and (Liang et al., 2020) optimize for class-specific filters, which are neither suitable for the low-dimensional case when the number of classes exceeds the number of features nor interpretable, since it is unclear if the feature is already detecting the class rather than a lower level feature. The Feature Redundancy Loss (Zheng et al., 2020) (FRL) enforces the $K$ most used features to be localized differently by reducing the normalized inner product between their feature maps. This adds a hyperparameter and does not optimize all features at once.

2.2 Interpretable Machine Learning

Interpretable machine learning is a broad term and can refer to both models that are interpretable by design, and post-hoc methods that try to understand what the model has learned. Furthermore, interpretability can be classified as the interpretability of a single instance (local) or the entire model (global) (Molnar, 2020).

In this work, we present methods making models more interpretable by design but also utilize post-hoc
methods to offer local and global interpretability. Common local post-hoc methods are saliency maps like Grad-CAM (Selvaraju et al., 2017) that aim to show what part of the input image is relevant for the prediction. While they can be helpful, they have to be cautiously interpreted, as they do not show many desired properties one would expect from an explanation like shift invariance (Kindermans et al., 2019) or only producing reasonable explanations, when the model is working as intended (Adebayo et al., 2018). Another way of obtaining saliency maps is based on masking the input image and measuring the impact on the output (Zeiler and Fergus, 2014; Fong and Vedaldi, 2017; Jain et al., 2022).

As a global post-hoc method, Elude (Ramaswamy et al., 2022) generates an explanation for a model by mimicking its behavior with a sparse model. This model uses additional attributes and main directions of the remaining feature space of the model as input. Instead of explaining a model, we directly train the more interpretable model in this work. Another line of research tries to align learned representations with human understandable concepts from an additional labeled dataset (Kim et al., 2018; Bau et al., 2017; McGrath et al., 2021), increasing the global interpretability of the model. While algorithms may aim to make the models decision more interpretable (e.g. applications of fuzzy models in game AI (Dockhorn and Kruse, 2021)), to the best of our knowledge, measuring the interpretability of a deep neural network is an open task, as previous work focuses on measuring the quality of explanations of black boxes (Rokade and Alluri, 2021) or on conventional machine learning algorithms (Islam et al., 2020), where increased interpretability is measured when model complexity is reduced, e.g. via the number of operations (Yang et al., 2017; Friedler et al., 2019; Rüping et al., 2006) or number of features (Rüping et al., 2006). The sparsity and low-dimensionality of our proposed SLDD-Model is motivated by these findings. Due to the limitations of post-hoc methods in explaining a deep neural network, models that are more interpretable by design are becoming more relevant. Prototype based models like ProtoTree (Nauta et al., 2021), ProtoNet (Chen et al., 2019), ProtoShare (Rymarczyk et al., 2021) and ProtoPool (Rymarczyk et al., 2022) combine a deep feature extractor with a more interpretable model by using the similarities to learned prototypes as input to the linear model or decision tree. While they achieve competitive performance on fine-grained image classification, (Kim et al., 2021) and (Hoffmann et al., 2021) indicate a gap between the perceived similarities of humans and prototype based models. Concept bottleneck models (CBM) (Koh et al., 2020) first predict concepts annotated in the dataset and then use a simple model to predict the target class from the concepts. CBM-AUC (Sawada and Nakamura, 2022) extended CBM by allowing unsupervised concepts to influence the decision. PCBM (Yaksekgonul et al., 2022) created a post-hoc CBM using TCAV (Kim et al., 2018) to compress high-dimensional learned features into a concept bottleneck. Margelou et al. (2021) and Elude (Ramaswamy et al., 2022) both suggest that training the CBM end-to-end leads to the encoding of additional information next to the concepts, which reduces the interpretability. In contrast to CBM, our proposed method does not require additional labels for training and leads to a very sparse decision process. While their features are generally more relevant. Prototype based models like ProtoTree (Nauta et al., 2021), ProtoNet (Chen et al., 2019), ProtoShare (Rymarczyk et al., 2021) and ProtoPool (Rymarczyk et al., 2022) combine a deep feature extractor with a more interpretable model by using the similarities to learned prototypes as input to the linear model or decision tree. While they achieve competitive performance on fine-grained image classification, (Kim et al., 2021) and (Hoffmann et al., 2021) indicate a gap between the perceived similarities of humans and prototype based models. Concept bottleneck models (CBM) (Koh et al., 2020) first predict concepts annotated in the dataset and then use a simple model to predict the target class from the concepts. CBM-AUC (Sawada and Nakamura, 2022) extended CBM by allowing unsupervised concepts to influence the decision. PCBM (Yaksekgonul et al., 2022) created a post-hoc CBM using TCAV (Kim et al., 2018) to compress high-dimensional learned features into a concept bottleneck. Margelou et al. (2021) and Elude (Ramaswamy et al., 2022) both suggest that training the CBM end-to-end leads to the encoding of additional information next to the concepts, which reduces the interpretability. In contrast to CBM, our proposed method does not require additional labels for training and leads to a very sparse decision process. While their features are generally more relevant. Prototype based models like ProtoTree (Nauta et al., 2021), ProtoNet (Chen et al., 2019), ProtoShare (Rymarczyk et al., 2021) and ProtoPool (Rymarczyk et al., 2022) combine a deep feature extractor with a more interpretable model by using the similarities to learned prototypes as input to the linear model or decision tree. While they achieve competitive performance on fine-grained image classification, (Kim et al., 2021) and (Hoffmann et al., 2021) indicate a gap between the perceived similarities of humans and prototype based models. Concept bottleneck models (CBM) (Koh et al., 2020) first predict concepts annotated in the dataset and then use a simple model to predict the target class from the concepts. CBM-AUC (Sawada and Nakamura, 2022) extended CBM by allowing unsupervised concepts to influence the decision. PCBM (Yaksekgonul et al., 2022) created a post-hoc CBM using TCAV (Kim et al., 2018) to compress high-dimensional learned features into a concept bottleneck. Margelou et al. (2021) and Elude (Ramaswamy et al., 2022) both suggest that training the CBM end-to-end leads to the encoding of additional information next to the concepts, which reduces the interpretability. In contrast to CBM, our proposed method does not require additional labels for training and leads to a very sparse decision process. While their features are generally more relevant. Prototype based models like ProtoTree (Nauta et al., 2021), ProtoNet (Chen et al., 2019), ProtoShare (Rymarczyk et al., 2021) and ProtoPool (Rymarczyk et al., 2022) combine a deep feature extractor with a more interpretable model by using the similarities to learned prototypes as input to the linear model or decision tree. While they achieve competitive performance on fine-grained image classification, (Kim et al., 2021) and (Hoffmann et al., 2021) indicate a gap between the perceived similarities of humans and prototype based models. Concept bottleneck models (CBM) (Koh et al., 2020) first predict concepts annotated in the dataset and then use a simple model to predict the target class from the concepts. CBM-AUC (Sawada and Nakamura, 2022) extended CBM by allowing unsupervised concepts to influence the decision. PCBM (Yaksekgonul et al., 2022) created a post-hoc CBM using TCAV (Kim et al., 2018) to compress high-dimensional learned features into a concept bottleneck. Margelou et al. (2021) and Elude (Ramaswamy et al., 2022) both suggest that training the CBM end-to-end leads to the encoding of additional information next to the concepts, which reduces the interpretability. In contrast to CBM, our proposed method does not require additional labels for training and leads to a very sparse decision process. While their features are generally more relevant. Prototype based models like ProtoTree (Nauta et al., 2021), ProtoNet (Chen et al., 2019), ProtoShare (Rymarczyk et al., 2021) and ProtoPool (Rymarczyk et al., 2022) combine a deep feature extractor with a more interpretable model by using the similarities to learned prototypes as input to the linear model or decision tree. While they achieve competitive performance on fine-grained image classification, (Kim et al., 2021) and (Hoffmann et al., 2021) indicate a gap between the perceived similarities of humans and prototype based models. Concept bottleneck models (CBM) (Koh et al., 2020) first predict concepts annotated in the dataset and then use a simple model to predict the target class from the concepts. CBM-AUC (Sawada and Nakamura, 2022) extended CBM by allowing unsupervised concepts to influence the decision. PCBM (Yaksekgonul et al., 2022) created a post-hoc CBM using TCAV (Kim et al., 2018) to compress high-dimensional learned features into a concept bottleneck. Margelou et al. (2021) and Elude (Ramaswamy et al., 2022) both suggest that training the CBM end-to-end leads to the encoding of additional information next to the concepts, which reduces the interpretability. In contrast to CBM, our proposed method does not require additional labels for training and leads to a very sparse decision process. While their features are generally more relevant.
aim is understanding the neural network, they fit the sparse layer to fixed features and do not finetune the features to the sparse layer, which requires different optimization strategies than dense networks (Tessera et al., 2021). In this work, we utilize glm-saga to create a more interpretable model with competitive accuracy by applying it on selected features with higher diversity and finetuning the features afterwards. This leads to improved accuracy and enables a higher sparsity which, combined with the reduced number of features, increases interpretability.

3 Method

3.1 Problem Formulation

We apply the proposed SLDD-Model to the domain of fine-grained image classification. We consider the problem of classifying an image $I \in \mathbb{R}^{T \times w \times h}$ of width $w$ and height $h$ into one class $c \in \{c_1, c_2, \ldots, c_n\}$ using a trainable deep neural network $\Phi$. This neural network extracts the feature maps $M \in \mathbb{R}^{n_f \times w_M \times h_M}$ and aggregates them into the feature vector $f \in \mathbb{R}^{n_f}$. Then it applies the trainable neural network $C$ to obtain the final output $y \in \mathbb{R}^{n_c}$ as $y = C(f)$.

3.2 SLDD-Model

\begin{figure}[h]
  \centering
  \includegraphics[width=\textwidth]{pipeline.png}
  \caption{Overview of our proposed pipeline to construct a SLDD-Model.}
\end{figure}

We propose a flexible, generally applicable method for generating a more locally and globally interpretable model with no need for additional labels and an adjustable tradeoff between interpretability and accuracy. We make the decision process more interpretable by only using $n_f^s$ features with $n_f^s \ll n_f$ and using an interpretable classifier $C$. At the core of an interpretable classifier $C$ lies a linear layer $y = Wf + b$ with the weight matrix $W \in \mathbb{R}^{n_c \times n_f^s}$ and bias $b \in \mathbb{R}^{n_c}$. In order for it to be interpretable, $W$ has to be very sparse, meaning the number of non-zero entries $n_w$ has to be very low. Miller (1956) showed that humans can handle $7 \pm 2$ cognitive aspects at once, which constitutes an appropriate upper bound on the average number of relevant features per class $n_{w^{c}} = \frac{n_f}{n_c}$. In our work we focus on $n_{w^{c}} \leq 5$.

The pipeline of our approach is presented in Figure 2 and utilizes glm-saga for sparsification and feature selection. We first train a deep neural network with our proposed feature diversity loss $L_{\text{div}}$ until convergence. Then the features $F_{\text{train}} \in \mathbb{R}^{n_f \times n_f}$ for all $n_T$ images in the training set are computed, which are the average pooled feature maps $M$. Afterwards, the features are selected as described in Section 3.2.2 and glm-saga, presented in Section 2.2.1, is used to calculate the regularization path. Finally, the solution with the desired sparsity is selected from the regularization path and the remaining layers get finetuned with the final layer set to the sparse model, s.t. the features adapt to it.

3.2.1 Feature Diversity Loss

The goal of the proposed feature diversity loss $L_{\text{div}}$ is that every feature captures a different independent concept. This is achieved by enforcing differently localized features in their feature maps. The proposed loss is motivated by the Mutual-Channel Loss (MCL) (Chang et al., 2020) and the Feature Redundancy loss (FRL) (Zheng et al., 2020). In contrast to FRL, we use Cross-Channel-Max-Pooling (CCMP) (Goodfellow et al., 2015) over all weighted feature maps to optimize all features jointly and reduce the need for the hyperparameter $K$. MCL also uses CCMP but instead of grouping the channels into class-specific filters, we apply the diversity component to all feature maps $M$ to aim for shared interpretable features. For notation, $w_{ij}$ describes the entry in $W$ that is assigned to the specific feature $l \in \{0, 1, \ldots, n_f - 1\}$ for the predicted class $c = \arg \max(y)$ and $m_{ij} = M_{i,j}^c$.

\begin{equation}
\hat{s}_{ij}^{l} = \frac{\exp(m_{ij}^l)}{\sum_{i'=1}^{b_M} \sum_{j'=1}^{w_M} \exp(m_{ij'}^{l'})} \max f \left( \left\| w_c \right\|_2 \right
\end{equation}

\begin{equation}
L_{\text{div}} = - \sum_{i=1}^{b_M} \sum_{j=1}^{w_M} \max(\hat{s}_{ij}^{1}, \hat{s}_{ij}^{2}, \ldots, \hat{s}_{ij}^{n_f})
\end{equation}
Model is discussed and these attributes are used to show the alignment of the learned features. The \( L \) after the competitive accuracy and the impact of \( \lambda \) for selecting the set of features is the learning rate of \( \lambda \). Additionally, \( CUB-2011 \) contains labels for the images such as attributes (\( \text{Aircraft} \) (Maji et al., 2013), \( \text{NABirds} \) (Van Horn et al., 2015) and \( \text{ImageNet-1K} \) is given in Table 1.

This section contains our experimental results. We validate our method using \( \text{Resnet50} \) (He et al., 2015). The resulting proximal operator with \( \lambda \) decreases the loss if the different weighted feature maps do not have to be localized in small background patches. Equation 4 indicates the importance of \( W \) and accuracy is visualized.

Equation 3 uses the softmax to transform the feature maps \( M \) by normalizing their entries \( m_{ij} \) over the spatial dimensions and then scales the maps so that they focus on visible and important features while weighting them according to the predicted class, s. t. \( f \) different to MCL absent features do not have to be localized in small background patches. Equation 4 decreases the loss if the different weighted feature maps \( S^l \) attend to different locations. The final training loss is then \( \mathcal{L}_{\text{final}} = \mathcal{L}_{\text{CE}} + \beta \mathcal{L}_{\text{div}} \) with the weighting factor \( \beta \in \mathbb{R}_+ \).

### 3.2.2 Feature Selection

For selecting the set of features \( N_f^* \) from the initial features \( N_f \) s. t. \( |N_f^*| = n_f^* \), we run an adapted version of \( \text{glm-saga} \), introduced in Section 2.2.1 until one solution \( ( W^{\text{sparse}}, b_j ) \) of the regularization path uses a feature not already in \( N_f^* \), which we then add to the set of selected features \( N_f^* \) and restart the adapted \( \text{glm-saga} \). As adaptation, we extended the proximal operator of the group version of \( \text{glm-saga} \), which operates on \( w_l = W_{-l,:} \) which are the entries in \( W \) that correspond to an entire feature \( l \). Since \( \| w_l \|_2 \) indicates the importance of \( l \), we additionally only keep entries for features that have the maximum norm or are in \( N_f^* \), s. t. exactly one feature is added per iteration. The resulting proximal operator with \( \lambda_1 = \gamma \lambda_\alpha \) and \( \lambda_2 = \gamma \lambda(1-\alpha) \) is:

\[
\text{Prox}_{\lambda_1, \lambda_2}(w_l) = \begin{cases} 
\frac{w_l((\|w_l\|_2 - \lambda_1)/(1+\lambda_2))\|w_l\|_2}{\|w_l\|_2 - \lambda_1} & \text{if } \|w_l\|_2 > \lambda_1 \land \|w_l\|_2 = \max_{j' \in N_f \setminus N_f^*} \|w_{j'}\|_2 \lor i \notin N_f^* \\
0 & \text{otherwise}
\end{cases}
\]

(5)

The extensions are underlined and \( \gamma \) is the learning rate of \( \text{glm-saga} \).

### 4 Experiments

This section contains our experimental results. We validate our method using \( \text{Resnet50} \) (He et al., 2016), \( \text{DenseNet121} \) (Huang et al., 2017) and Inception-v3 (Szegedy et al., 2016) on four common benchmark datasets in the domain of fine-grained image classification. Additionally, we show the applicability of a \( \text{SLDD-Model} \) for large scale datasets like ImageNet-1K (Russakovsky et al., 2015). An overview of \( \text{CUB-2011} \) (Wah et al., 2011), \( \text{Stanford Cars} \) (Krause et al., 2013), \( \text{FGVC-Aircraft} \) (Maji et al., 2013), \( \text{NABirds} \) (Van Horn et al., 2015) and \( \text{ImageNet-1K} \) is given in Table 1. Additionally, \( \text{CUB-2011} \) contains labels for the images such as attributes (e.g. “red wing”) as well as for the classes, which makes it easier to measure the alignment with understandable concepts. After the competitive accuracy and the impact of \( \mathcal{L}_{\text{div}} \) is shown, the interpretability of the \( \text{SLDD-Model} \) is discussed and these attributes are used to show the alignment of the learned features. The implementation details can be found in Appendix B.1. Finally, the tradeoff between interpretability and accuracy is visualized.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CUB-2011</th>
<th>Stanford Cars</th>
<th>FGVC-Aircraft</th>
<th>NABirds</th>
<th>ImageNet-1K</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Classes ( n_c )</td>
<td>200</td>
<td>196</td>
<td>100</td>
<td>555</td>
<td>1000</td>
</tr>
<tr>
<td>Training</td>
<td>5,994</td>
<td>8,144</td>
<td>6,667</td>
<td>23,929</td>
<td>1,281,167</td>
</tr>
<tr>
<td>Testing</td>
<td>5,774</td>
<td>8,041</td>
<td>3,333</td>
<td>24,633</td>
<td>50,000</td>
</tr>
</tbody>
</table>

Table 1: Overview of the number of classes, training and testing examples for the used datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>Additional Supervision required</th>
<th>Features trained end-to-end</th>
<th>( n_f^* )</th>
<th>( \mathcal{L}_{\text{div}} )</th>
<th>ImageNet-1K</th>
<th>CUB-2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBM (Koh et al., 2020) - joint</td>
<td>✓</td>
<td>✓</td>
<td>112</td>
<td>112</td>
<td>-</td>
<td>80.1</td>
</tr>
<tr>
<td>CBM (Koh et al., 2020) - independent</td>
<td>✓</td>
<td>x</td>
<td>112</td>
<td>112</td>
<td>-</td>
<td>76.0</td>
</tr>
<tr>
<td>CBM-AUC (Sawada and Nakamura, 2022)</td>
<td>✓</td>
<td>x</td>
<td>256</td>
<td>256</td>
<td>-</td>
<td>82.3</td>
</tr>
<tr>
<td>SLDD-Model (Ours) with ( n_f^* )</td>
<td>x</td>
<td>✓</td>
<td>5</td>
<td>76.7*</td>
<td>80.3</td>
<td></td>
</tr>
<tr>
<td>SLDD-Model (Ours)</td>
<td>x</td>
<td>✓</td>
<td>50</td>
<td>72.8*</td>
<td>78.3</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Comparison with competitors on accuracy in percent. For ease of comparison, we evaluated \( \text{CUB-2011} \) on Inception-v3 \((n_f = 1024) \) and \( \text{ImageNet-1K} \) with \( \text{Resnet50} \) \((n_f = 2048) \) (*denotes \( \text{Resnet50} \) accuracy). The dense \( \text{Resnet50} \) achieves 80.9% on \( \text{ImageNet-1K} \). For \( \text{glm-saga} \), we selected the solution with maximum \( n_f^* \leq 5 \). Arrows indicate generally preferable directions. For \( \text{CBM - joint} \), \( \text{Resnet50} \) results were created as described in Appendix B.1.1.
With diversity@k

Table 3 shows the competitive performance of our approach to competitors: Without finetuning the DenseNet121 with n_w = 2048, the model with finetuned features, obtained by training the sparse model on the training data, and still n_w = 5. Every shown metric is the average over five (four for ImageNet-1K) randomly seeded runs. The standard deviations are included in the appendix.

Table 3: Impact of the loss function on accuracy in percent for Resnet50. Best results are in bold.

4.1 Diversity Metric

To assess the impact of \( \mathcal{L}_{\text{div}} \), we developed a measurement for the local diversity of the feature maps \( M \) that led to the decision, inspired by the diversity component of MCL (Chang et al., 2020), which entails a different way of computing the features. For that, we consider the \( k \) feature maps \( M_k \) that are weighted the highest for the predicted class \( c \) in \( W \). To only compare the localization, softmax is applied to the \( M_k \), yielding \( S_k \). With these distributions \( S_k \), we compute the diversity as

\[
\text{diversity@} k = \frac{\sum_{i=1}^{h_k} \sum_{j=1}^{w_k} \max(s_{ij}^1, s_{ij}^2, \ldots, s_{ij}^k)}{k} \tag{6}
\]

with diversity@\( k \in \left[ \frac{1}{k}, 1 \right] \) to measure how different and pronounced the \( M_k \) are localized. Since we focus on \( n_w \geq 5 \), we set \( k = 5 \). We report the mean diversity@5 for all classes that use at least five features. Note that the proposed \( \mathcal{L}_{\text{div}} \) is a weighted version of diversity@\( n_f \).

4.2 Results

We report the accuracy on the test set for the dense model after training the pretrained model on the training data with \( n_w = n_f \), for the sparse model with \( n_w \leq 5 \), and for the result of our whole pipeline, the model with finetuned features, obtained by training the sparse model on the training data, and still \( n_w \leq 5 \). Every shown metric is the average over five (four for ImageNet-1K) randomly seeded runs. The standard deviations are included in the appendix.

Table 4 shows the competitive performance of our SLDD-Model to the dense Resnet50. It is evident that an extreme sparsity of \( s = \frac{5}{2048} \) can be obtained in the final layer with just 0.1 to 0.4 percent points less accuracy. Additionally decreasing the number of features by 97.6%, resulting in just 50 instead of the previous 2048 features, only reduces the accuracy compared to the dense model by 1.3 to 2.7 percent points. Finally, our proposed \( \mathcal{L}_{\text{div}} \) improves the accuracy for all sparse models. Table 4 shows the general applicability of our method with different backbones. However, we observed some instability and no increased accuracy when finetuning the DenseNet121 with \( n_f = 2048 \), showing a positive effect of sharing features. Table 2 compares our approach to competitors: Without requiring additional supervision, we achieve a competitive performance compared to CBM (Koh et al., 2020)-based methods, while achieving a lower dimensionality and higher sparsity. Additionally, we improve the accuracy of gbm-sage with heavily reduced \( n_f \). For ImageNet-1K, we skipped the dense training and directly used the pretrained model. The good scalability of our proposed method to this large dataset with a higher number of classes is displayed in Table 5. Table 5 shows the diversity@5: With \( \mathcal{L}_{\text{div}} \), it is very close to the maximum value of 100% in the dense case and still heavily increased in the sparse cases. This showcases that \( \mathcal{L}_{\text{div}} \) is suitable to ensure a diverse localization and improved interpretability of the used feature maps, which is visualized in Figures [9] to [17]. Finally, the total number of features that is used by the unrestricted (\( n_f \)) models in Table 5 is reduced from 912 to 719 for the models with \( \mathcal{L}_{\text{div}} \), but still shows a high number of class-specific features. That \( \mathcal{L}_{\text{div}} \) leads to more shared features supports our motivation of enforcing different features to capture different concepts.

Table 4: Accuracy in percent dependent on backbone. Best results are in bold.

Table 5: Accuracy in percent for Resnet50 on ImageNet-1K using the pretrained dense model.
L\_div

<table>
<thead>
<tr>
<th></th>
<th>CUB-2011</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(n_f)</td>
<td>Dense</td>
<td>Sparse</td>
<td>Finet.</td>
<td>Sparse</td>
<td>Finet.</td>
<td>Sparse</td>
<td>Finet.</td>
</tr>
<tr>
<td>2048</td>
<td>50.2</td>
<td>46.0</td>
<td>41.4</td>
<td>48.0</td>
<td>46.5</td>
<td>46.8</td>
<td>45.9</td>
</tr>
<tr>
<td>50</td>
<td>46.5</td>
<td>43.8</td>
<td>40.9</td>
<td>49.9</td>
<td>44.0</td>
<td>45.0</td>
<td>41.7</td>
</tr>
<tr>
<td></td>
<td>45.0</td>
<td>41.7</td>
<td>39.1</td>
<td>43.6</td>
<td>43.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Impact of the loss function on diversity@5 in percent for Resnet50. Best results are in bold.

4.2.1 Comparison with Other Loss Functions

We compare our diversity loss \(L\_div\) with the MCL (Chang et al., 2020) and the FRL (Zheng et al., 2020). The used hyperparameters for the loss functions are reported in Appendix B.1.1 and we focused on three datasets to save computational resources. Table 3 shows that our \(L\_div\) reaches the highest accuracy across all datasets. Notably, the accuracy reported in (Chang et al., 2020) for the MC-Loss is achieved by a two-layer MLP plus additional techniques, whereas we only use one layer to ensure linearly separable representations for our SLDD-Model. Although it is expected that applying \(L\_div\) has a positive effect on diversity@5 due to the similar formulation, we could observe a remarkable uplift in diversity@5 (Table 6) compared to MCL and FRL, which also optimize for differently localized features.

4.3 Interpretability

In this section, we discuss the interpretability of the proposed SLDD-Model using example models. The interpretability of the proposed SLDD-Model is based upon using very few \(n_f\) features from a small pool of \(n_f\) to make a decision. A low \(n_f\) allows the analysis of the remaining features to try to align them with a human understandable concept, which is discussed in Section 4.3.1. Since the sparse linear layer is easily interpretable, the complete model with sufficiently well understood features is both locally and globally interpretable.

For global interpretability, the final layer of the SLDD-Model can be fully visualized and analyzed. Figure 3a shows \(W\_sparse\). This allows the practitioner to verify the global behavior of the model. For example, the attribute aligned with “has-bill-shape:needle” in the presented model in Section 4.3.1 has a non-zero weight for all four classes that have the attribute in more than 30 % of examples. The visualization of the classes positively related to a specific attribute and features related to a class like Figures 9 to 17 helps to trust the model. Additionally, \(W\_sparse\) allows for further feature understanding, since it is possible to analyze the similarities between classes that share a feature. If the feature is aligned well, this leads to knowledge discovery.

The local interpretability describes the explanation of a single decision made by the model. Decisions with sparsely connected features are inherently locally interpretable, if the features can be interpreted and localized, as shown in Figure 1. The practitioner can understand where and what was found in the image, and due to the full global interpretability also understand the behavior around the current example.
4.3.1 Feature Alignment
In this section, we demonstrate how the features of the proposed SLDD-Model can be aligned with interpretable concepts. We describe how one can use additional labels or expert knowledge to interpret the features and demonstrate that several learned sparse features are directly connected to attributes relevant to humans. Thus, our model learns such abstract concepts directly from the data. Overall, due to the very limited number of used features $n_f$, the features can and should be thoroughly analyzed and interpreted to facilitate interpretability. For feature localization, we follow a masking approach similar to Long and Vedaldi (2017), which is described in Appendix D.

Alignment with Additional Data We use the attributes $A$ contained in CUB-2011 to align the learned features with these labels after the finetuning. For each attribute $a \in A$ and feature $j$ we compute a score $C_{aj}$ that corresponds to a relative increase of the feature when the attribute is present:

$$
\delta_{aj} = \frac{1}{|\rho_{a+}|} \sum_{i \in \rho_{a+}} F_{i,j}^{\text{train}} - \frac{1}{|\rho_{a-}|} \sum_{i \in \rho_{a-}} F_{i,j}^{\text{train}}
$$

$$
C_{aj} = \frac{\delta_{aj}}{\max(F_{:,j}^{\text{train}}) - \min(F_{:,j}^{\text{train}})}
$$

The set of indices whose images contain the attribute are denoted by $\rho_{a \pm}$. We considered an attribute to be present if the human annotated it with “probably” or “definitely”. Annotations with “guessing” were neither included in the positive ($\rho_{a+}$) nor negative ($\rho_{a-}$) examples. For one exemplary model a part of the matrix of $C$ values is displayed in Figure 3b. It is clear, that some features correspond to colors, some to specific shapes like “bill-shape:needle” and other features do not correlate with specific attributes. Figure 4a visually validates the connection that was implied in Figure 3b.

Manual Alignment To ensure that the entirety of a feature is understood, or in absence of additional data, the features have to be manually aligned for increased interpretability. This is enabled by the low number of features and sparsity. Some useful aspects for understanding a feature are the localization, extreme examples, feature visualization (Olah et al., 2017) or which classes use that feature. One such alignment for a model trained on FGVC-Aircraft is displayed in Appendix E and Figure 4b. Aligning learned features with human understandable concepts is still challenging as a single feature can refer to multiple aspects and human understandable concepts do not need to be axis aligned (Szegedy et al., 2013). However, the low dimensionality of the remaining features allows for a sophisticated analysis of every feature in practice, which could even discover spurious correlations as features as done by the glm-saga publication.

4.4 Interpretability Tradeoff
In this section, we analyze the impact of changing $n_f$ and $n_{wc}$ on the finetuning accuracy of the model trained with $L_{div}$, shown in Figure 5. Figure 5a visualizes the impact of $n_f$ on accuracy. With decreasing $n_f$ the accuracy drops slowly until a dataset-specific threshold is reached, at which a steep decline
starts. Additionally, the proposed $L_{\text{div}}$ works regardless of $n_f^*$. Figure 5b shows the finetuning accuracy in relation to $n_{wc}$: The accuracy is rather insensitive to $n_{wc}$, only decreasing when $n_{wc} < 5$, which is the case for both dimensions and showcases that five features suffice for a competitive model even if the features are shared among classes. Figure 5 demonstrates the tradeoff that our SLDD-Model offers: Both $n_{wc}$ and $n_f^*$ can be drastically reduced with either a negligible or small impact on accuracy to adapt to the amount of interpretability needed.

## 5 Limitations and Future Work

As shown in Figure 5, the SLDD-Model cannot get arbitrarily low-dimensional or sparse with competitive accuracy via our proposed method. The optimal sparsity and dimensionality for a given problem are hard to predict and might require some experiments to determine the minimum values for competitive accuracy. Aligning all used features with human concepts is still difficult, albeit more feasible than without a SLDD-Model. Future work could use a more interpretable feature extractor like B-cos Networks (Böhle et al., 2022) to alleviate that problem. The method could be improved via the feature selection or improvements for small datasets (Reinders et al., 2022). It seems promising to apply a SLDD-Model to other safety-critical domains, such as medical, where an expert can be utilized to align the features and follow the decision, as it can help bring the required interpretability and trustworthiness to the domain. Embodied autonomous agents can also benefit from it, as the entire decision process can be thoroughly analyzed. In this domain, our approach could be extended to anomaly detection (Rudolph et al., 2022) in addition to classification. While more interpretable models could be used to more deliberately bring harm, they can disclose existing problems with machine learning models and open up the opportunity to build fair and trustworthy models. Finally, sparsity and dimensionality could be part of metrics used to quantify the trustworthiness of a model.

## 6 Conclusion

In this work, we proposed the more interpretable sparse low-dimensional decision model (SLDD-Model) to allow a human to follow and understand the decision of a Deep Neural Network for image classification. Our proposed pipeline constructs a SLDD-Model with drastically increased global and local interpretability while still showing competitive accuracy. As demonstrated, a practitioner can manually configure the pipeline to set the tradeoff between accuracy and interpretability. Our novel loss increases the feature diversity and we showed that identifying a class with varied features can improve the accuracy. Finally, our SLDD-Model offers measurable aspects of interpretability, which allows future work to not just compare itself on accuracy but also on interpretability.

**Acknowledgements.** This work was supported by the Federal Ministry of Education and Research (BMBF), Germany under the project Leibniz/KILabor (grant no. 01DD20003), the Center for Digital Innovations (ZDIN) and the Deutsche Forschungsgemeinschaft (DFG) under Germany’s Excellence Strategy within the Cluster of Excellence PhoenixD (EXC 2122).
References


Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In 4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13), Sydney, Australia, 2013.


George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological review, 63(2):81, 1956.


Decomposition in unlabelled / labelled features. Sparse explanation, not prediction as it uses ground truth attribute labels for explanation. Does not have to be faithful with actually done prediction.


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes] See Section 4.2, more specifically Table 12
   (b) Did you describe the limitations of your work? [Yes] See Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See section 5
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The data is missing as it is freely available. The code is not included but the method and instructions are quite detailed.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [No] We mentioned everything except how they were chosen.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See standard deviations in Appendix
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] Total figure is not available

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section B.1
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]
Table 7: Accuracy in percent dependent on $L_{div}$ for Resnet50. Best results per column are in bold and ± indicates the standard deviation across five runs.

<table>
<thead>
<tr>
<th>Table 8: Accuracy in percent dependent on backbone. Best results per column are in bold and ± indicates the standard deviation across five runs.</th>
</tr>
</thead>
</table>

A Appendix

In this appendix, we provide implementation details and standard deviations for the experiments. Additionally, the pseudocode for glm-saga [Wong et al., 2021] is shown. Finally, we present the feature visualization technique and more ablations on $L_{div}$.

B Detailed Results

The full results of Section 4.2 with the standard deviations are presented in Tables 7 to 13. The reported standard deviations are, except for DenseNet121 on NABirds, as mentioned in Section 4.2, generally rather small compared to the differences in means, which supports our conclusions. We also show exemplary images with the 5 most important features for the dense and sparse conventional model in Figures 9 to 17. The comparison with the finetuned SLDD-Model shows an improved localization and interpretability of the features. The finetuned SLDD-Model in Figure 10 seems to use features that each individually focus more on chest, lower belly, head, bill or crown, whereas for the dense and sparse models the different features focus on the same regions. This increased diversity@5 and with it interpretability was also measured in Section 4.2.

B.1 Implementation Details

We use Pytorch [Paszke et al., 2019] to implement our methods and on ImageNet pretrained models as backbone feature extractor. We utilized glm-saga and robustness [Engstrom et al., 2019]. The images are resized to $448 \times 448$ (299×299 for Inception-v3, 224×224 for ImageNet-1K), normalized, randomly horizontally flipped and jitter is applied. The model is finetuned using stochastic gradient descent on the specific dataset for 150 (100 for NABirds) epochs with a batch size of 16 (64 for ImageNet-1K), starting with $5 \times 10^{-4}$ as learning rate for the pretrained layer and 0.01 for the final linear layer. Both get multiplied by 0.4 every 30 epochs. Additionally, we used momentum of 0.9, $\ell_2$-regularization of $5 \times 10^{-4}$ and apply a dropout rate of 0.2 on the features to reduce dependencies. $\beta$ was set to 0.196 for Resnet50, 0.098 for DenseNet121 and 0.049 for Inception-v3. For the feature selection, we set $\alpha = 0.8$ and reduce the regularization strength $\lambda$ by 90% as we found it sped up the process without decreasing performance.

We use glm-saga to compute the regularization path with $\alpha = 0.99$ and all other parameters set to default with a lookbehind of $T = 5$. From this path, the solution with maximum $n_{agg} \leq 10$ is selected. Then the non-zero

<table>
<thead>
<tr>
<th>Table 9: Accuracy in percent for Resnet50 on ImageNet-1K using the pretrained dense model. Best results per column are in bold and ± indicates the standard deviation across four runs.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ImageNet-1K</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>$n_f^* = 2048$</th>
<th>$n_f^* = 50$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dense</td>
<td>Sparse</td>
</tr>
<tr>
<td>80.9</td>
<td>62.2</td>
</tr>
</tbody>
</table>
Algorithm 1 Pseudocode from glm-saga \cite{Wong et al. 2021}

\begin{algorithm}
\begin{algorithmic}[1]
  \State Initialize table of scalars $a'_i = 0$ for $i \in [n]$
  \State Initialize average gradient of table $g_{avg} = 0$ and $g_{0avg} = 0$
  \For{$i \in B$}
    \State $a_i = x_i \beta + \beta_0 - y_i$
    \State $g_i = a_i \cdot x_i$ \Comment{calculate new gradient information}
    \State $g'_i = a'_i \cdot x_i$ \Comment{calculate stored gradient information}
  \EndFor
  \State $g = \frac{1}{|B|} \sum_{i \in B} g_i$
  \State $g' = \frac{1}{|B|} \sum_{i \in B} g'_i$
  \State $g_0 = \frac{1}{|B|} \sum_{i \in B} a_i$
  \State $g'_0 = \frac{1}{|B|} \sum_{i \in B} a'_i$
  \State $\beta = \beta - \gamma (g - g' + g_{avg})$
  \State $\beta_0 = \beta_0 - \gamma (g_0 - g'_0 + g_{0avg})$
  \State $\beta = \text{Prox}_{\lambda_2, \lambda_1} (\beta) $
  \For{$i \in B$}
    \State $a'_i = a_i$ \Comment{update table}
    \State $g_{avg} = g_{avg} + \frac{|B|}{n} (g - g')$ \Comment{update average}
    \State $g_{0avg} = g_{0avg} + \frac{|B|}{n} (g_0 - g'_0)$
  \EndFor
\end{algorithmic}
\end{algorithm}

entries with the lowest absolute value get zeroed out until we are left with $\nu = 5$, as we empirically found that they do not improve test accuracy after finetuning.

This selected solution replaces the final layer of our model. Then we train for 40 epochs, starting with the final learning rate of the initial training multiplied by 100 ($\frac{1}{100}$ of that for ImageNet-1K), and decrease it by 60% every 10 epochs. Dropout on the features was set to 0.1 and momentum was increased to 0.95. Note that, while the increased momentum has been important for the stability of the final training, the hyperparameters were not thoroughly optimized for the sparse case.

B.1.1 Competitors

For creating the accuracy for Resnet50 and CBM \cite{Koh et al. 2020} - \textit{joint} in Table 2, we resized the images to $448 \times 448$ and used a batch size of 16. The remaining used hyperparameters were almost identical to the CBM experiments with Inception-v3, but we only trained for up to 400 epochs, as 650 led to decreased accuracy (~0.8 percent points). Additionally, the learning rate was not decayed, mirroring the published code. The reported accuracy stems from three runs with a standard deviation of 0.7.

For MCL, we used the reported hyperparameters of $\mu = 0.005$ and $\lambda = 10$. For finetuning, we assigned every feature to every class that was using it. We optimized the hyperparameters for FRL based on accuracy, leading to $K = 10$ and $\lambda = 0.01$.

C Glm-saga

This section includes the Pseudocode for glm-saga \cite{Wong et al. 2021} in algorithm 1. The proximal operator $\text{Prox}_{\lambda_1, \lambda_2} (\beta)$ is defined as:

$$\text{Prox}_{\lambda_1, \lambda_2} (\beta) = \begin{cases} 
\frac{\beta - \lambda_1}{1 + \lambda_2} & \text{if } \beta > \lambda_1 \\
\frac{\beta + \lambda_1}{1 + \lambda_2} & \text{if } \beta < \lambda_1 \\
0 & \text{otherwise}
\end{cases} \quad (8)$$

D Visualization of Features

For feature visualization, we follow a masking approach. We systematically blur, following \cite{Fong and Vedaldi 2017}, one patch of size $p \times p$ of the image and measure the difference in feature activation between the augmented image and not augmented image. The actual localization map $L_p \in \mathbb{R}^{n_7 \times \frac{p}{8} \times \frac{p}{8}}$ for that square size
Whether all correlating concepts have to be understood or how exact the analysis has to be depends on the application.

Table 10: diversity@5 in percent dependent on $L_{div}$ for Resnet50. Best results per column are in bold and ± indicates the standard deviation across five runs.

Table 11: diversity@5 in percent dependent on $L_{div}$ on backbone. Best results per column are in bold and ± indicates the standard deviation across five runs.

is computed by

$$L_{p_{xy}} = \text{ReLU}(f(x) - f(p_{xy}))$$  \hspace{1cm} (9)

where $p_{xy}$ indicates the image where a $p$-sized patch starting at position $(x, y)$ is blurred and the ReLU suppresses parts that increased the feature activation, since blur should not be injecting a feature. The final localization map is the combination of different square sizes $p \in \{28, 56, 64, 112, 224\}$ to accommodate for differently sized features:

$$L = \sum_p \frac{L_p}{\max(L_p)} \hspace{1cm} (10)$$

Notably, $L_p$ has to be resized according to the smallest $p$ and we only show $L'$ for one feature $i$.

### E Feature Alignment

In this section, we use the alignment of the shown feature in Figure 4b with four-engine aircraft to exemplary show how one can align a feature manually. We first visualize the distribution of the feature over the training data in Figure 8. This indicates a more binary attribute and by investigating the images and saliency maps, we observed an alignment with four-engine aircraft. We test the hypothesis by filtering for the classes “A340-500” and “BAE 146-300”, for which the feature corresponds to the four engines. Figure 7 shows that the lowest activating examples of this group do not clearly show the four-engines which supports our hypothesis. Note that one feature can correspond to multiple concepts, as another hypothesis is an alignment with the propeller. Whether all correlating concepts have to be understood or how exact the analysis has to be depends on the application.

### F Ablations on Feature Diversity Loss

In this section, we present an additional analysis of the factors in $L_{div}$ and the impact of $\beta$.

Table 12: Accuracy in percent for Resnet50 compared to other loss functions. Best results per column are in bold and ± indicates the standard deviation across five runs.
Figure 6: Top: Distribution of feature activation over the training data. Bottom: Different examples of this distribution with their scaled feature localization. The activation and class name is given above the image.

F.1 Factor Importance

We analyzed the impact of the two factors in Equation 3 with for accuracy optimized \( \beta \), shown in Tables 14 and 15. The label \textit{w/o Class-Specific} indicates not using the weights of the predicted class and \textit{w/o Rescaling} refers to not maintaining their relative mean. We used \( \beta = 0.001 \cdot \frac{196}{2048} \approx 1e-5 \) for \textit{w/o Class-Specific}, since we use the size of the feature maps with \( 196 = w_M \cdot h_M \) and the number of features of the baseline model \( n_f = 2048 \) as scaling factors in order to be less dependent of model architecture and image size, and \( \beta = 0.1 \) for \textit{w/o Rescaling}. Only the combination of both factors leads to an improved accuracy, validating our idea that it is important to only enforce diversity of features that are found in the input and used in conjunction.
Figure 7: The three least activating training examples of classes "A340-500" and "BAE-16-300" for the given feature.

Table 13: diversity@5 in percent for Resnet50 compared to other loss functions. Best results per column are in bold and ± indicates the standard deviation across five runs.

<table>
<thead>
<tr>
<th>Loss</th>
<th>CUB-2011 n\textsubscript{f} = 2048</th>
<th>CUB-2011 n\textsubscript{f} = 50</th>
<th>FGVC-Aircraft n\textsubscript{f} = 2048</th>
<th>FGVC-Aircraft n\textsubscript{f} = 50</th>
<th>Stanford Cars n\textsubscript{f} = 2048</th>
<th>Stanford Cars n\textsubscript{f} = 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o Class-Specific</td>
<td>96.0</td>
<td>84.0</td>
<td>86.5</td>
<td>81.7</td>
<td>84.0</td>
<td>91.4</td>
</tr>
<tr>
<td>w/o Rescaling</td>
<td>±0.1</td>
<td>±0.2</td>
<td>±0.1</td>
<td>±0.2</td>
<td>±0.3</td>
<td>±0.2</td>
</tr>
<tr>
<td>w/o Class-Specific</td>
<td>86.3</td>
<td>82.7</td>
<td>85.6</td>
<td>80.2</td>
<td>83.4</td>
<td>90.8</td>
</tr>
<tr>
<td>w/o Class-Specific</td>
<td>±0.3</td>
<td>±0.5</td>
<td>±0.7</td>
<td>±0.3</td>
<td>±0.5</td>
<td>±0.4</td>
</tr>
</tbody>
</table>

Table 14: Impact of factors in \( L_{\text{div}} \) on accuracy in percent for Resnet50. Best results per column are in bold and ± indicates the standard deviation across five runs.

<table>
<thead>
<tr>
<th>Loss</th>
<th>CUB-2011 n\textsubscript{f} = 2048</th>
<th>CUB-2011 n\textsubscript{f} = 50</th>
<th>FGVC-Aircraft n\textsubscript{f} = 2048</th>
<th>FGVC-Aircraft n\textsubscript{f} = 50</th>
<th>Stanford Cars n\textsubscript{f} = 2048</th>
<th>Stanford Cars n\textsubscript{f} = 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o Rescaling</td>
<td>98.9</td>
<td>69.9</td>
<td>71.9</td>
<td>65.2</td>
<td>72.6</td>
<td>98.8</td>
</tr>
<tr>
<td>w/o Rescaling</td>
<td>±0.1</td>
<td>±0.3</td>
<td>±0.4</td>
<td>±1.4</td>
<td>±0.3</td>
<td>±0.2</td>
</tr>
<tr>
<td>w/o Class-Specific</td>
<td>99.3</td>
<td>67.6</td>
<td>68.9</td>
<td>70.5</td>
<td>72.3</td>
<td>99.7</td>
</tr>
<tr>
<td>w/o Class-Specific</td>
<td>±0.0</td>
<td>±0.9</td>
<td>±0.5</td>
<td>±2.0</td>
<td>±1.8</td>
<td>±0.0</td>
</tr>
<tr>
<td>w/o Class-Specific</td>
<td>50.6</td>
<td>46.2</td>
<td>43.5</td>
<td>48.1</td>
<td>46.8</td>
<td>47.6</td>
</tr>
<tr>
<td>w/o Class-Specific</td>
<td>±0.3</td>
<td>±0.3</td>
<td>±0.2</td>
<td>±1.1</td>
<td>±0.6</td>
<td>±0.6</td>
</tr>
</tbody>
</table>

Table 15: Impact of factors in \( L_{\text{div}} \) on diversity@5 in percent for Resnet50. Best results per column are in bold and ± indicates the standard deviation across five runs.
β with a slight increase of β. For the feature diversity loss \( L \) value higher leads to F.2 Loss Weighting supports our approach of enforcing varied features for the extremely sparse case. classification, we choose across all datasets in the sparse case with increasing proposed method, we use \( \beta \) indicates the standard deviation across five runs. Our used
\[
\begin{array}{cccccccc}
\text{F.2 Loss Weighting} & \text{supports our approach of enforcing varied features for the extremely sparse case.}
\end{array}
\]

Table 16: Accuracy in percent dependent on \( \beta \) for Resnet50. Best results per column are in bold and ± indicates the standard deviation across five runs. Our used \( \beta \) is underlined.

<table>
<thead>
<tr>
<th>( \beta )</th>
<th>Dense</th>
<th>Sparse</th>
<th>Finet</th>
<th>Dense</th>
<th>Sparse</th>
<th>Finet</th>
<th>Dense</th>
<th>Sparse</th>
<th>Finet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>86.6</td>
<td>81.8</td>
<td>85.3</td>
<td>79.5</td>
<td>83.4</td>
<td>90.0</td>
<td>88.4</td>
<td>89.4</td>
<td>87.3</td>
</tr>
<tr>
<td>±0.4</td>
<td>±0.3</td>
<td>±0.2</td>
<td>±0.3</td>
<td>±0.2</td>
<td>±0.3</td>
<td>±0.2</td>
<td>±0.4</td>
<td>±0.3</td>
<td>±0.1</td>
</tr>
<tr>
<td>0.00196</td>
<td>86.4</td>
<td>82.0</td>
<td>85.5</td>
<td>79.1</td>
<td>83.3</td>
<td>90.2</td>
<td>88.7</td>
<td>89.5</td>
<td>87.6</td>
</tr>
<tr>
<td>±0.2</td>
<td>±0.3</td>
<td>±0.3</td>
<td>±0.3</td>
<td>±0.2</td>
<td>±0.2</td>
<td>±0.3</td>
<td>±0.3</td>
<td>±0.2</td>
<td>±0.2</td>
</tr>
<tr>
<td>0.0196</td>
<td>86.4</td>
<td>82.2</td>
<td>85.3</td>
<td>79.4</td>
<td>83.2</td>
<td>90.6</td>
<td>89.0</td>
<td>89.7</td>
<td>87.4</td>
</tr>
<tr>
<td>±0.2</td>
<td>±0.3</td>
<td>±0.3</td>
<td>±0.6</td>
<td>±0.3</td>
<td>±0.4</td>
<td>±0.3</td>
<td>±0.4</td>
<td>±0.4</td>
<td>±0.1</td>
</tr>
<tr>
<td>0.196</td>
<td>86.6</td>
<td>84.0</td>
<td>86.5</td>
<td>81.7</td>
<td>84.0</td>
<td>91.4</td>
<td>90.7</td>
<td>91.1</td>
<td>89.8</td>
</tr>
<tr>
<td>±0.2</td>
<td>±0.2</td>
<td>±0.1</td>
<td>±0.2</td>
<td>±0.3</td>
<td>±0.2</td>
<td>±0.3</td>
<td>±0.2</td>
<td>±0.4</td>
<td>±0.1</td>
</tr>
</tbody>
</table>

Table 17: diversity@5 in percent dependent on \( \beta \) for Resnet50. Best results per column are in bold and ± indicates the standard deviation across five runs. Our used \( \beta \) is underlined.

<table>
<thead>
<tr>
<th>( \beta )</th>
<th>Dense</th>
<th>Sparse</th>
<th>Finet</th>
<th>Dense</th>
<th>Sparse</th>
<th>Finet</th>
<th>Dense</th>
<th>Sparse</th>
<th>Finet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>50.2</td>
<td>46.0</td>
<td>43.4</td>
<td>48.0</td>
<td>46.5</td>
<td>46.5</td>
<td>43.8</td>
<td>48.9</td>
<td>45.9</td>
</tr>
<tr>
<td>±0.2</td>
<td>±0.2</td>
<td>±0.3</td>
<td>±0.5</td>
<td>±0.3</td>
<td>±0.5</td>
<td>±0.5</td>
<td>±0.7</td>
<td>±0.4</td>
<td>±0.6</td>
</tr>
<tr>
<td>0.00196</td>
<td>51.0</td>
<td>46.1</td>
<td>43.7</td>
<td>49.2</td>
<td>47.6</td>
<td>48.1</td>
<td>44.6</td>
<td>41.5</td>
<td>45.6</td>
</tr>
<tr>
<td>±0.1</td>
<td>±0.2</td>
<td>±0.2</td>
<td>±0.6</td>
<td>±0.4</td>
<td>±0.4</td>
<td>±0.4</td>
<td>±0.4</td>
<td>±0.4</td>
<td>±0.5</td>
</tr>
<tr>
<td>0.0196</td>
<td>60.0</td>
<td>50.8</td>
<td>47.8</td>
<td>50.2</td>
<td>48.6</td>
<td>75.4</td>
<td>54.4</td>
<td>50.2</td>
<td>50.2</td>
</tr>
<tr>
<td>±0.6</td>
<td>±0.6</td>
<td>±0.4</td>
<td>±1.0</td>
<td>±0.9</td>
<td>±0.5</td>
<td>±0.7</td>
<td>±0.6</td>
<td>±1.9</td>
<td>±1.0</td>
</tr>
<tr>
<td>0.196</td>
<td>98.9</td>
<td>69.9</td>
<td>71.9</td>
<td>65.2</td>
<td>72.6</td>
<td>98.8</td>
<td>85.7</td>
<td>86.6</td>
<td>69.3</td>
</tr>
<tr>
<td>±0.1</td>
<td>±0.5</td>
<td>±0.4</td>
<td>±1.4</td>
<td>±0.3</td>
<td>±0.2</td>
<td>±1.5</td>
<td>±1.4</td>
<td>±0.8</td>
<td>±1.3</td>
</tr>
</tbody>
</table>

F.2 Loss Weighting

This section is concerned with the impact of the weighting factor \( \beta \) for the feature diversity loss \( L_{div} \). For our proposed method, we use \( \beta = 0.196 \). Tables [16] and [17] show that \( L_{div} \) improves the diversity@5 and accuracy across all datasets in the sparse case with increasing \( \beta \) up to a maximum roughly around \( \beta = 1 \). Setting the value higher leads to \( L_{div} \) dominating the training. To ensure that the network is still mainly optimized for classification, we choose \( \beta = 0.196 \), even though in some cases we could still observe a slight gain in accuracy with a slight increase of \( \beta \). The positive relation between diversity@5 and accuracy, visualized in Figure 8 supports our approach of enforcing varied features for the extremely sparse case.
Figure 8: Relationship between finetuned diversity@5 and accuracy for varying $\beta$ for Resnet50, portrayed via color, on FGVC-Aircraft. Each dot represents an increase by a factor of $\sqrt{10}$ and the standard deviation is indicated by the shaded area. $\beta = 1.96 \ L_{\text{div}}$ is not shown, as it dominates the training.

Figure 9: Feature maps of the top 5 features by magnitude for class Scarlet Tanager on example images. The used weights for the respective features are also displayed.
Figure 10: Feature maps of the top 5 features by magnitude for class White Crowned Sparrow on example images. The used weights for the respective features are also displayed.

Figure 11: Feature maps of the top 5 features by magnitude for class Brown Creeper on example images. The used weights for the respective features are also displayed.
Figure 12: Feature maps of the top 5 features by magnitude for class Cape Glossy Starling on example images. The used weights for the respective features are also displayed.

Figure 13: Feature maps of the top 5 features by magnitude for class Ford Freestar Minivan on example images. The used weights for the respective features are also displayed.
Figure 14: Feature maps of the top 5 features by magnitude for class Audi S5 Coupe 2012 on example images. The used weights for the respective features are also displayed.

Figure 15: Feature maps of the top 5 features by magnitude for class Ford Ranger SubCab2011 on example images. The used weights for the respective features are also displayed.
Figure 16: Feature maps of the top 5 features by magnitude for class Tornado on example images. The used weights for the respective features are also displayed.

(a) Conventional Dense Model  
(b) Sparse with $n^*_f = n_f$  
(c) Finetuned SLDD-Model

Figure 17: Feature maps of the top 5 features by magnitude for class Fokker 100 on example images. The used weights for the respective features are also displayed.

(a) Conventional Dense Model  
(b) Sparse with $n^*_f = n_f$  
(c) Finetuned SLDD-Model