UNIVERSITAT DE BARCELONA

UNIVERSITAT ROVIRA i VIRGILI

Master's Thesis

Master in Artificial Intelligence

Facultat d'Informàtica de Barcelona (FIB)

Facultat de Matemàtiques (UB)

Escola Tècnica Superior d'Enginyeria (URV)

Exploring Multi-Subset Learning Techniques for Fine-Grained Food Image Classification

---

Author:
Pablo Villacorta Benito

Advisor:
Dr. Petia Radeva
Dept. Matemàtiques i Informàtica (UB)

Co-advisor:
Javier Ródenas Cumplido

Date of defense: January 25, 2023

Abstract

Fine-grained image recognition (FGIR) is a fundamental and challenging problem within the field of computer vision that involves analyzing visual objects from subordinate categories, such as bird species or car models. The applications of FGIR are plentiful in both industry and research, ranging from automatic biodiversity monitoring to intelligent transportation. Recent advances in deep learning have paved the way for significant progress in this field. A recently proposed method is FGFR [51], a food-centered fine-grained recognition method that leverages a multi-task architecture in which different heads or tasks specialize in discriminating between classes of automatically detected subsets of hard-to-distinguish classes. In this work, we provide an in-depth analysis of the behavior of FGFR and propose an improved version, FGFR+, which builds on top of the limitations we identify from our study of the original method.

While we prove that FGFR is capable of generalizing to other non-food domains and different types of backbone architectures, we also observe that the method is not taking full advantage of its specialized multi-head structure. We find that, by implementing a series of conceptually simple modifications, the performance of the method can be significantly boosted, capitalizing on the fine-grained knowledge provided by the heads. FGFR+ achieves 94.2% top-1 validation accuracy on the Food-101 dataset, virtually ranking third in its corresponding benchmark. Being compatible with a wide range of deep learning computer vision backbone architectures, FGFR+ could have the potential of boosting the performance of many computer vision classification tasks.

Keywords
Fine-grained recognition, subsets of classes, multi-task learning

Contents

Chapter 1

Introduction

Fine-grained image recognition constitutes a fundamental problem within the field of computer vision. Recent breakthroughs in computer-based image analysis have demonstrated that machines can outperform human capabilities on tasks that rely on visual data, especially when dealing with coarse-grained objects. However, addressing fine-grained visual objects still remains a difficult problem. While the average human might effortlessly notice that a bronzed cowbird can be distinguished form a shiny cowbird by the color of its eyes, teaching machines to locate and emphasize on such specific discriminative regions of an image still remains an open challenge.

As a consequence, significant progress has been done in recent years in the field of fine-grained image recognition, in an attempt of improving machines' capabilities to differentiate between hard-to-distinguish classes. Different strategies have been proposed to tackle this problem, such as using a two-step method that first localizes the informative regions and then makes a prediction based on them. Other modern approaches try to learn how to effectively encode the important discriminative information into image features.

A particularly challenging domain within the field of fine-grained image analysis, and computer vision in general, is food. Food dishes are inherently unstructured objects that can present large intra-class variations which can make a single type of dish have many different visual appearances. Another common, well-known issue is the difficulty of properly annotating food images, as a single dish may belong to many different classes (different ingredients) and some ingredients may present a hierarchical structure (having classes such as spaghetti and pasta) that may not always be reflected in the annotations.

In this work, we focus on a recently published method for Fine-Grained Food Recognition (FGFR) that was proposed in [51]. This work introduces a novel multi-step approach for performing fine-grained food image classification, leveraging a multi-task architecture. FGFR is capable of automatically identifying subsets of classes that are hard to distinguish, and builds a multi-head model where each head specializes in discriminating between the classes within its corresponding subset. Using this approach, the authors of [51] managed to outperform multiple state-of-the-art approaches for different fine-grained food datasets.

The method, however, was introduced as a preliminar approach that was only evaluated on a limited set of datasets, all of them belonging to the food domain. Thus, in this work, we aim at both evaluating the generalization ability of FGFR to different visual domains and model architectures, as well as providing a more insightful understanding of its behavior. Based on our analysis, we also identify potential areas of improvement for the original method, and we propose a series of upgrades based on them, introducing the method which we refer to as FGFR+. We evaluate the extended method against the original one, and demonstrate its superiority in a variety of domains, achieving a performance that is competitive with current state-of-the-art approaches for at least one of our evaluation datasets.

1.1 Objectives

The primary objectives of our project can be outlined as follows:

- **Objective 1:** perform an extensive study of the original FGFR, analyzing each of its steps individually and assessing its ability to generalize to different domains and model architectures.
- **Objective 2:** based on the analysis of the original method, identify potential limitations or areas of improvement and propose modifications that alleviate them.
- **Objective 3:** provide a detailed analysis of the behavior of each of the proposed modifications to determine their contribution to the performance of the method.

1.2 Contributions

The contributions of our work are summarized as follows:

1. We demonstrate that the original FGFR is capable of generalizing to different domains (i.e., birds, cars and food), as well as to different backbone architectures (i.e., EfficientNet [55] and CSWin Transformer [13]).
2. We provide an in-depth analysis of the behavior of the original FGFR, carefully evaluating the results obtained after each of its steps and identifying potential areas of improvement.
3. We propose a series of improvement proposals based on our study of the original method, under the name of FGFR+, which allow to fully unlock the potential of the multi-head model.
4. We show the ineffectiveness of integrating some seemingly reasonable approaches into the FGFR method, namely the use of self-attention as a way of combining the multi-head outputs.
5. We demonstrate that FGFR+ consistently outperforms the original FGFR in a variety of dataset domains and backbone architectures.

Our proposed method managed to achieve a top-1 validation accuracy of 94.2% for the Food-101 dataset [6], which, to our best knowledge, would rank in third place for the Food-101 fine-grained image classification benchmark, among those methods that do not use additional training data.

1.3 Document Structure

This document is organized in the following way. Chapter 2 provides an overview of the most relevant related work to our project within the fields of fine-grained image recognition and multi-task learning. Chapter 3 describes in more detail both the original FGFR, as well as some other complementary techniques that were used throughout the development of our work. Our proposal for FGFR+ is introduced in Chapter 4. Chapter 5 reports the evaluation process and results of both the original and the FGFR+ methods. More concretely, this chapter includes a description of the evaluation strategy and settings followed, a report of the study of the original FGFR, the results obtained using FGFR+, and an analysis of some of the most relevant aspects we experimented with while developing our proposal. Finally, in Chapter 6, we gather the conclusions we reached during the development of our project and discuss some other relevant aspects of it, while also pointing out some promising lines of future work.

Chapter 2

State of the Art

Focusing on the problem of fine-grained image recognition, in this chapter, we introduce this field and describe some of the related works within them that we found to be relevant for both the original FGFR and our extended version, FGFR+. We also introduce the field of multi-task learning and highlight some of the relevant related work within this area.

2.1 Fine-Grained Image Recognition

Within Computer Vision, the field of Fine-Grained Image Recognition (FGIR) aims at recognizing images that belong to different subordinate categories of a meta-category. Some classical examples of fine-grained image recognition problems include the discrimination between different subspecies of animals, car models or food dishes. These types of images tend to have a very similar global appearance, but present some key differences in some subtle (or fine-grained) features. Therefore, the main challenge within fine-grained image analysis lies on correctly identifying and modelling the key discriminative regions of a given image.

Fine-grained image analysis lies in the continuum between basic-level category analysis and instance-level analysis (as illustrated in Figure 2.1). Basic-level category analysis refers to the common problem of generic image analysis (e.g., discriminating between a frog and a piano), whereas instance-level analysis deals with the problem of discriminating between different individuals (such as identifying whether two pictures of a human correspond to the same person).

More concretely, the main difference between fine-grained and generic image analysis is the fact that in generic image analysis, the different classes belong to basic-level meta-categories that are coarse-grained and, consequently, present clear visual differences. In fine-grained image analysis, however, object categories belong to a single meta-category, which inherently makes their appearances visually similar.

Fine-grained image analysis is a particularly hard problem given that images tend to present small inter-class variations (as a consequence of having very similar sub-categories), and large intra-class variations (accounting for different poses, angles and scales). In basic-level image analysis, on the other hand, images present small intra-class variations and large inter-class variations, which

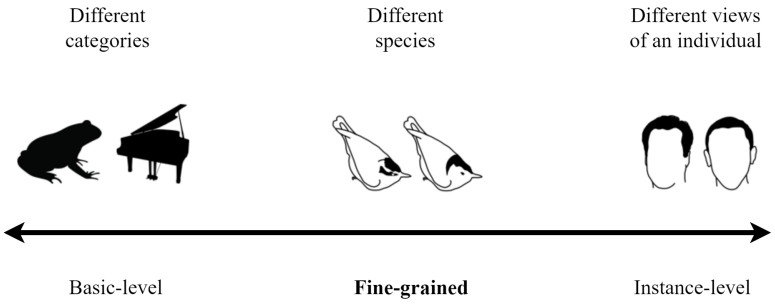


Figure 2.1: Illustration of the continuum in which fine-grained analysis lies between basic-level and instance-level analysis.

facilitates the discrimination between different classes.

Most FGIR methods can be grouped into two main paradigms [62]: 1) recognition by localization-classification subnetworks, and 2) recognition by end-to-end feature encoding.

Recognition by localization-classification subnetworks is a historically popular approach that deals with the problem of locating and classifying the key discriminative semantic parts of fine-grained objects. The key idea of the concept is to use two separate subnetworks. A localization subnetwork will locate the key parts of an object and will generate a local part-level feature vector for each of the parts. Typically, these methods also generate a global object-level feature vector. A classification subnetwork will then receive both the local and global representations and will perform recognition.

Different approaches vary in the way they model the localization subnetwork. Some methods employ detection or segmentation techniques to locate relevant semantic image regions [19, 35, 24] (an example method is shown in Figure 2.2). Other approaches based on convolutional neural networks (CNNs) utilize the activations from different convolutional layers as localized descriptors, which can be linked to semantic parts of common objects [65]. Some relevant approaches that make use of CNN filter outputs as part detectors can be found in [39, 61, 29]. Attention mechanisms have also been used to detect key object parts even in those cases where it might be hard to define common parts of some objects. This is specially relevant for non-structured objects, such as food dishes. In these cases, attention mechanisms can be used so that the model can attend to loosely defined regions of fine-grained objects. Relevant works in this direction include [16, 48, 67].

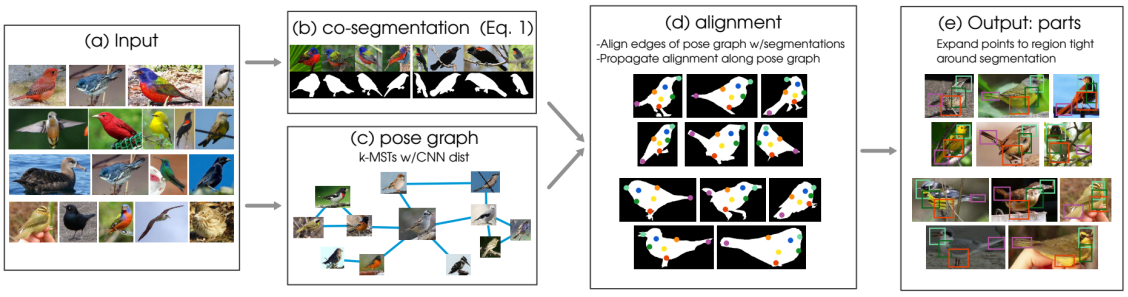


Figure 2.2: Diagram illustrating the method introduced in [35]. The method relies on co-segmentation [20] to obtain the object masks, and then aligns images with similar poses in order to extract common parts of objects. Figure borrowed from [35].

Recognition by end-to-end feature encoding is a paradigm of FGIR that focuses on feature learning, which plays an important role in fine-grained problems as it helps capture and differentiate between the subtle differences between different sub-categories. Different methods have been developed to learn single, discriminative image representations using approaches such as high-order feature interactions [15] or designing specific loss functions [14, 15]. Multiple methods showed the effectiveness of integrating the covariance matrix-based representation with deep descriptors, such as [38], where an image is represented as a pooled outer product of features extracted from two CNNs. The approach proposed in [45] follows a different idea, maximizing the entropy of the output probability distribution, following the intuition that it might be reasonable to prevent a fine-grained classifier from being too confident in its output, discouraging low entropy.

Current state-of-the-art methods for fine-grained datasets would also fall under this paradigm. The authors in [3] introduce a novel context-aware attentional pooling (CAP) approach. CAP is able to pick up on subtle variations by using sub-pixel gradients, and it can learn to focus on relevant regions and determine their significance in differentiating fine-grained subcategories. They also propose a new method to represent features by taking into account the consistency between the informativeness of key regions and their spatial arrangements.

In [9], the authors propose a plug-in module (PIM) that can be implemented on top of many popular network architectures, such as CNNs or Transformers, to yield highly discriminative regions. PIM generates pixel-level feature maps and fuses filtered features with the goal of improving the fine-grained classification. The work [50] introduces CAL (Counterfactual Attention Learning), with which the authors propose using counterfactual causality to guide the learning process of visual attention, encouraging the network to learn more useful attention for the task of fine-grained image recognition.

2.1.1 Fine-Grained Image Recognition Based on Subset Learning

Here we introduce some existing works that are closely related to FGFR [51], tackling the problem of fine-grained visual recognition by extracting subsets of similar classes. The work [17] introduces a hierarchical subset learning approach, where the images of a dataset are clustered based on deep convolutional features. For each subset, they train a local support vector machine (SVM) classifier that discriminates between the classes of its subset, and they also train multiple classifiers to act as *subset selectors* so that, during inference, the system can decide which local SVM classifier to use to make the prediction for the given image. The method is illustrated in Figure 2.3. [18] introduces a similar approach, but learning features specific to each subset using separate subset-specific CNNs.

These works differ from FGFR [51] in multiple ways. On the one hand, they extract the class subsets by clustering visual representations of the images. In FGFR, instead, the clustering procedure is guided by model errors, by using the confusion matrix to determine the distances between classes. On the other hand, both [17] and [18] make use of subset selectors to determine which subset the current image belongs to, whereas FGFR relies on a fully connected (FC) layer to automatically determine the optimal way of combining the different classifiers.

It must also be noted that these approaches do not train an end-to-end model. Instead, they train individual classifiers that are then executed sequentially. FGFR, instead, replicates this process using a single model in a multi-task learning fashion, where the specialized classifiers are modelled

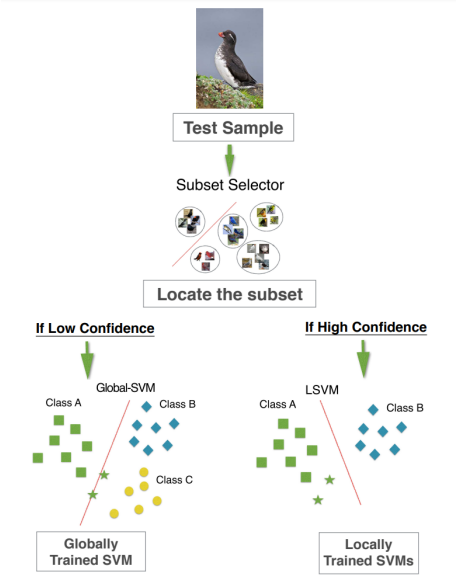


Figure 2.3: Overview of the method introduced in [17]. During inference, a subset selector is used to determine which specialized SVM will be in charge of making the final prediction. Figure borrowed from [17].

as parallel model heads with a shared backbone feature extractor.

2.2 Multi-Task Learning

Multi-task learning (MTL) is a paradigm in which a single model is trained to perform multiple tasks simultaneously. Traditionally, many MTL architectures follow the *shared trunk* outline [10], where a global feature extractor is shared by all tasks, followed by the individual task model heads that are in charge of making the task-specific predictions. Some examples of methods following this concept can be found in [66, 11, 43, 40]. An overview of TCDCN, the method introduced in [66] using this paradigm, is shown in Figure 2.4.

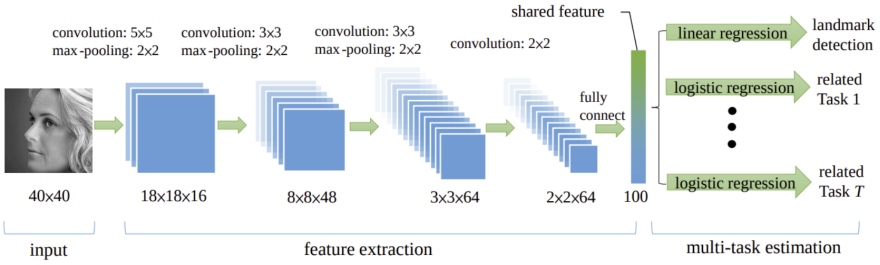


Figure 2.4: Diagram showing the multi-task architecture of TCDCN [66], one of the earliest works implementing the concept of the shared trunk using deep convolutional networks. Figure borrowed from [66].

Having to learn to perform multiple tasks simultaneously, the optimization of multi-task models is a crucial and challenging process. One of the most common and straightforward ways of modelling the multi-task loss is by formulating the final aggregated loss as a weighted sum of the task-specific losses. Correctly balancing the weight assigned to each task is fundamental to properly learn all

tasks. There exist multiple approaches for automatically learning the weights associated to the tasks. [32] was one of the earliest works to learn loss weights. In their work, the weight assigned to each task is dependent on its homoscedastic uncertainty, and the weights are parameters that are learned jointly with the model weights. [36] introduces an almost identical approach, modifying the regularization parameter to prevent the loss from becoming negative during training.

Some other methods weigh a task's loss depending on its learning speed, such as [8, 40]. Most of these methods increase a task's loss weight whenever its learning speed is low, in order to balance the learning between the tasks. [40] uses the ratio of a task' current loss and its previous loss as a way of measuring its learning speed. As an alternative to learning speed, other methods weigh the tasks' loss based on their performance [21, 30].

Chapter 3

Methodology

In this chapter, we explain in more detail the most relevant methods or techniques we used during the development of our project. In particular, we start by thoroughly describing the original FGFR as it was introduced in [51], and we continue by explaining some relevant models, methods and techniques that played an important role during our work, including the backbone architectures we used [55, 13], several multi-task loss functions [32, 36], self-attention [59] and focal loss [37].

3.1 The Original FGFR Method

This part of the document aims at introducing the fine-grained food recognition method proposed in [51], FGFR. The goal of the method is to improve the ability of an image classification model to correctly identify fine-grained classes.

FGFR is conceptually based on building on top of an already trained image classification model (the *baseline*) and applying a *classes subsets extraction* procedure to identify groups of classes that the baseline classifier tends to confuse with one another. Note that each of these subsets constitutes a fine-grained recognition problem of its own. Then, the original baseline is extended, replicating the last block of the original architecture (the *head*) as many times as there are subsets, and each replicated head will specialize in distinguishing the classes within its corresponding subset. The expanded model is trained in a multi-task way (where each subset has its own associated classification task), allowing each head to specialize in its subset. After the training of the heads, a fully connected (FC) layer is trained, to aggregate the predictions made by each of the heads (including the head of the original baseline) and generate a final prediction. Figure 3.1 illustrates the proposed scheme.

FGFR consists of three well-defined steps:

1. Baseline training and multi-subsets extraction.

2. Specialization step.

3. Aggregation step.

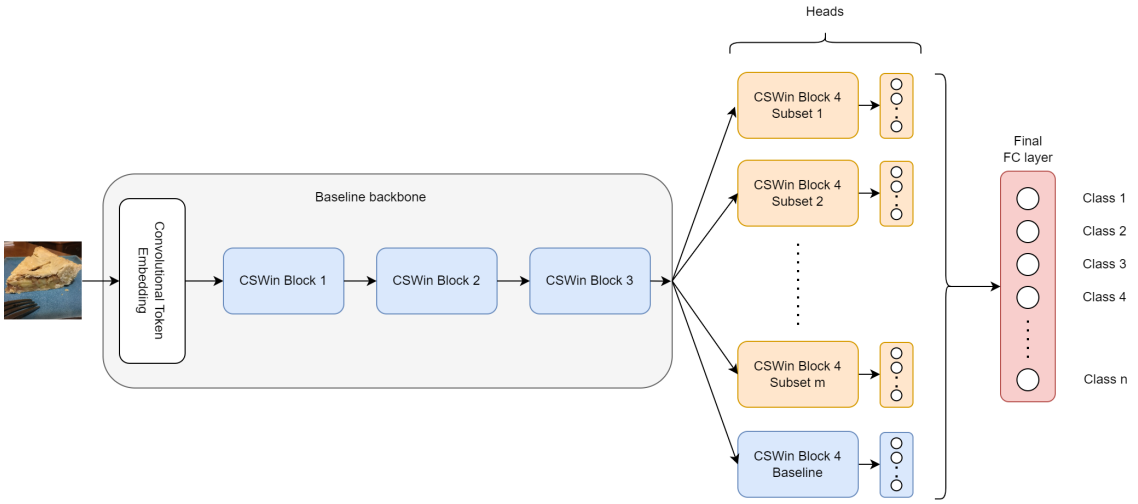


Figure 3.1: Diagram illustrating the complete model architecture during the aggregation step. The outputs of all heads are concatenated and passed to the final FC layer, which will aggregate all predictions and make the final classification. In the original FGFR, only the final FC layer (in red) is trained during this step.

Each of them will now be explained individually in more detail.

3.1.1 Step 1: Baseline Training and Multi-Subsets Extraction

The method starts with the training of a standard image classification model, as seen in Figure 3.2, the baseline. In order to extract the subsets of classes, a hierarchical clustering procedure is applied based on the confusion matrix obtained from the baseline model. The authors transform the confusion matrix into a distance matrix using the following formula, which defines the distance d between two classes c_i and c_j as:

$$d(c_i, c_j) = 1 - \frac{1}{2}(M_{ij} + M_{ji}), \tag{3.1}$$

where M represents the confusion matrix generated by the model. The authors normalize the confusion matrix so that the resulting distances are bound between 0 and 1. Once the distance between all pairs of classes is computed, an agglomerative clustering method is applied in order to obtain a hierarchy of class similarities. This dendrogram is then cut by applying a threshold value, in order to obtain the final clusters of classes. The hierarchical clustering is generated using single linkage, and the dendrogram threshold value is automatically calculated as the midpoint of the largest vertical difference between nodes.

3.1.2 Step 2: Specialization Step

After the multi-subsets extraction, the model needs to be extended so that it can have multiple parallel heads, each specializing in a different subset. In order to build the expanded model, the original baseline model needs to be split into two parts: the *backbone* and the *head*. The backbone will act as a generic feature extractor that will be shared by all the heads. The splitting point between the backbone and the head can be arbitrarily set, always assuming that the size of the head will be relatively small compared to that of the backbone. For instance, in the example

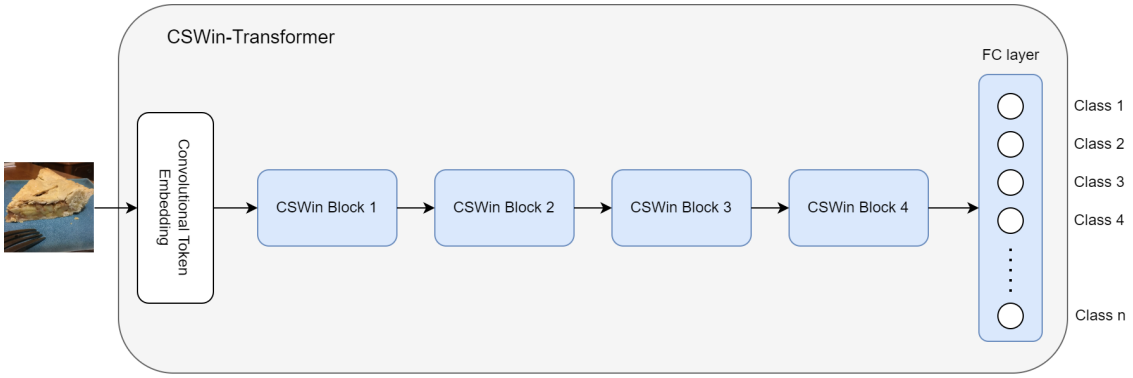


Figure 3.2: Diagram illustrating the different blocks that constitute a CSWin Transformer model, plus the final FC layer that performs the classification. Although here the example of CSWin is used, any other backbone architecture can be used, as long as it allows for a split between backbone and head.

used in the original paper [51], out of the four main transformer blocks that constitute the CSWin Transformer, the first three blocks are taken as the backbone, whereas the last remaining block is considered to be the head.

The head part of the model will be then replicated in parallel as many times as there are clusters. The final classification FC layer of each specialized head will be adapted so that it has as many outputs as there are classes in its corresponding cluster. Additionally, the classifiers of all heads (except for the original baseline head, which remains intact) will predict an additional class, 'others'. This class includes all the classes that do not belong to the specific subset, so that the classifier of a head learns to discriminate between the classes in its cluster and the rest of classes in the dataset. The result of this process is illustrated in Figure 3.3.

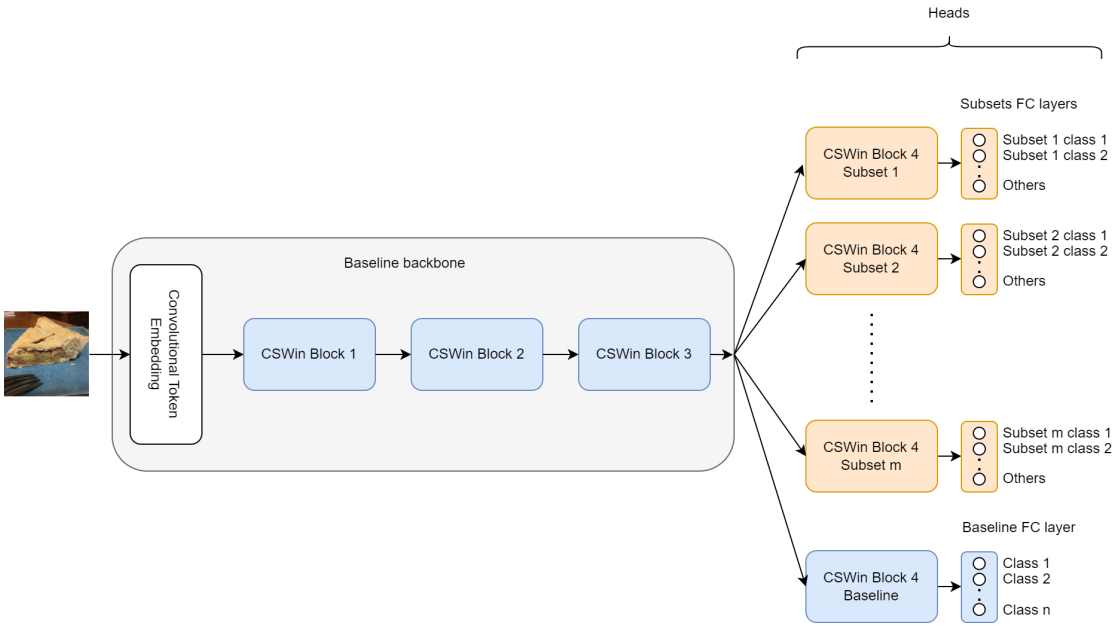


Figure 3.3: The multi-head CSWin model used for the specialization step. The shape of the classifier FC layer of each head is adapted to predict all classes in its subset plus the 'others' class. Only the specialized heads (in orange) are trained during the specialization step.

To train the second step of the pipeline, the *specialization* step, the authors propose keeping the weights of the baseline frozen (that is, the shared backbone and the original, generic head that makes predictions for all classes in the dataset), and only training the replicated specialized heads in a multi-task way, where each of the heads constitutes a task of its own. The multi-task training is simply performed by computing the cross-entropy loss of each head separately, and then averaging the loss of all heads to obtain a final loss value:

$$CE_i(y, \hat{y}) = -\sum_{j=1}^{N_{subset_i}} y_j log(\hat{y}_j), \mathcal{L} = \frac{1}{N_{subsets} + 1} \sum_{i=1}^{N_{subsets}+1} CE_i, \tag{3.2}$$

where y_j is the ground truth, $\hat{y}_j$ is the corresponding classifier prediction, CE_i is the cross-entropy of head i, CE_0 is the cross-entropy of the baseline head, and $N_{subsets}$ is the number of subsets.

Data balancing is crucial for this step of the method, as both intra-subset and inter-subset imbalance are likely to be present during the training of the heads. Inter-subset imbalance refers to the difference in the number of samples and classes present in each subset, whereas intra-subset imbalance refers to the different number of samples per class within a subset. The inclusion of the 'others' class within the specialized heads implies that there will be a strong intra-subset imbalance that needs to be handled.

To that end, FGFR tackles inter- and intra-subset imbalance individually. On the one hand, to fight inter-subset imbalance a method is proposed to determine the amount of samples per epoch L that are used to train each of the heads, which takes into account the number of classes n and the median number of samples per subset ms:

$$L_{subset} = \frac{n \cdot ms}{k}, \tag{3.3}$$

where k is a factor to avoid memory overload (the authors used $k = 8$). With this method, heads with a higher number of samples or classes will be trained with more samples, and vice versa. On the other hand, the problem of intra-subset imbalance is tackled by simply making a weighted random sampling of each image, based on the probability of occurrence of its class.

3.1.3 Step 3: Aggregation Step

In the final step, the outputs of the classifiers of all heads (both the specialized ones and that of the original baseline) are combined to generate a final prediction. The authors [51] propose using a fully connected layer to join the predictions coming from different heads. That is, the class predictions of all heads will be concatenated and passed to the final FC layer to perform the final prediction. In order to retain the classification performance of the baseline, the weights of the final FC layer are initialized with the identity matrix. Assuming that weight $w_{i,j}$ connects the output i from the baseline classifier with neuron j, $w_{i,j}$ is initialized to 1 if $i = j$, and 0 otherwise. In this way, when given an image that does not belong to any subset, the final classification will likely be the same as that of the original baseline. Regarding the weights connected to the outputs of the specialized heads, these are initialized randomly. During the training of this last step, which

just involves training the final FC layer while keeping the rest of the model frozen, the model will learn how to leverage the information provided by the specialized heads to outperform the original baseline. Figure 3.1 shows the complete resulting model.

FGFR managed to achieve validation results that outperformed the previous state-of-the-art on three public food recognition datasets containing fine-grained data: Food-101 [6], FoodX-251 [31] and Food1K (a sub-dataset built from Food2K [46]).

3.2 Baseline Model Architectures

It must be noted that FGFR is designed so that it can be integrated with many different deep learning computer vision models, such as CNNs or Vision Transformers (ViTs). In our work, we experiment with two different model architectures of notably different nature and size, in order to assess the generalization capability of FGFR. The two architectures are EfficientNet-B0 [55] and CSWin Transformer-Large [13]. We selected these backbone architectures given the remarkable results they achieved in the bibliography [55, 13].

EfficientNet [55] is a family of CNNs that was developed to improve both the efficiency and accuracy of image classification tasks. One of its most remarkable features is its compound scaling method, which allows to adjust the network's depth, width and resolution in a principled manner. Back when it was first published, the authors managed to achieve state-of-the-art results for a variety of datasets with an order of magnitude fewer parameters than the top-performing models. A diagram illustrating the architecture of EfficientNet models can be found in Figure 3.4.

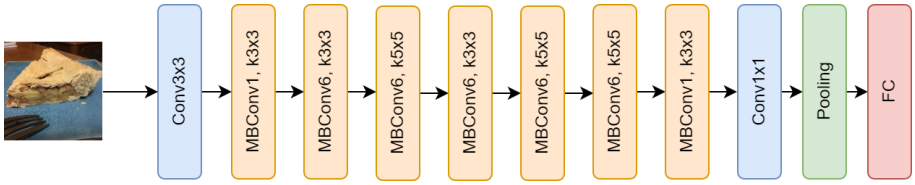


Figure 3.4: Diagram showing the structure of the generic EfficientNet [55] architecure.

The EfficientNet family includes a range of eight different models that vary in their scale (depth, width and resolution), ranging from the smallest version, EfficientNet-B0 (with around 5.3M parameters), to the largest one, EfficientNet-B7 (66M parameters). For our work, we make use of EfficientNet-B0, as we wanted to evaluate the FGFR method on a performant *CNN* model of relatively *small* size.

CSWin Transformer [13] is a recently introduced Transformer-based architecture designed for general-purpose vision tasks. The architecture addresses the challenge of balancing the trade-off between global self-attention, which is computationally expensive, and local self-attention, which limits the field of interactions of each token. To resolve this issue, the CSWin transformer employs a Cross-Shaped Window self-attention mechanism, which computes self-attention in parallel across horizontal and vertical stripes of equal width, forming a cross-shaped window. This approach leads to an efficient and effective design that can handle a wide range of vision tasks. The overall architecture of the CSWin transformer is shown in Figure 3.5.

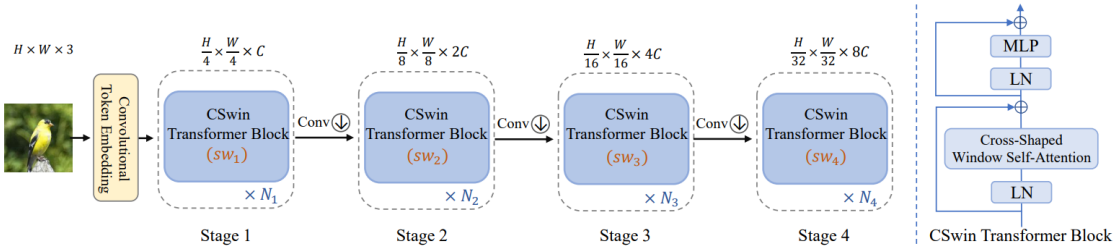


Figure 3.5: Overview of the CSWin transformer[13] architecture. Figure borrowed from [13].

The authors introduced multiple variants of the CSWin Transformer [13], which vary in their number of parameters: CSWin-T (Tiny, 23M parameters), CSWin-S (Small, 35M parameters), CSWin-B (Base, 78M parameters) and CSWin-L (Large, 173M parameters). For our work, we experiment with CSWin-L, given that it was also used in the original work of FGFR [51], and because we can use it as a counterpart of the EfficientNet-B0: a *transformer*-based model of relatively *big* size.

3.3 Multi-Task Learning Loss Functions

Given that the specialization step described in Section 3.1 is a crucial part of the method (as it is the part where the actual fine-grained learning takes place), we explore different ways of improving its training by leveraging existing MTL optimization methods. In particular, we focus on MTL optimization methods based on loss weighting, where the final loss value for a MTL problem is computed as a weighted sum of the losses of the individual tasks. For this work, we experimented with two popular approaches: weighting losses based on uncertainty [32], and the approach introduced in [36], which is presented as an improvement over the former.

3.3.1 Loss Weighting

The key idea behind multi-task loss weighting is to balance the individual loss functions for different tasks. Each task will have its own corresponding loss, which must be combined with those from the other tasks in order to compute a single aggregated loss function which the model is trained to optimize. Loss weighting methods parameterize the final aggregated loss function as a weighted sum of the losses of the individual tasks. Different approaches have been proposed to find an optimal set of weights for any given MTL problem.

One of the earliest and most well-known MTL loss weighting approaches was the one introduced in [32] (Uncertainty Weighted Loss, UWL, for brevity). Its authors argue that homoscedastic uncertainty can be used as a basis for weighting losses in a multi-task learning problem, treating the multi-task network as a probabilistic model, and deriving a weighted multi-task loss function by maximizing the likelihood of the ground truth output.

The resulting loss function jointly maximizes the likelihood of each of the task output distributions, and is defined as follows in the case of having two single-task cross-entropy classification losses $\mathcal{L}_1$ and $\mathcal{L}_2$:

$$\mathcal{L}(\mathbf{W}, \sigma_1, \sigma_2) = \frac{1}{\sigma_1^2}\mathcal{L}_1(\mathbf{W}) + \frac{1}{\sigma_2^2}\mathcal{L}_2(\mathbf{W}) + \log \sigma_1 + \log \sigma_2 \tag{3.4}$$

The model is trained to optimize both the parameters W of the network, as well as the weights σ_i of each task i. The loss function is carefully designed so that it is smoothly differentiable while also preventing that the weights converge to zero.

A similar loss weighing method is the one proposed by [36], which we refer to as AWL (Automatic Weighted Loss). The authors propose using a different regularization term for each task i, $R_i = \log(1 + c_i^2)$ in order to enforce positive regularization values. This prevents one of the main issues with the method proposed in [32], which could lead to negative loss values. The complete formulation for their loss weighting method is the following for classification tasks:

$$\mathcal{L}_{\mathrm{T}}(x, y_{\mathrm{T}}, y'_{\mathrm{T}}; w_{\mathrm{T}}) = \sum_{\tau \in \mathrm{T}} \frac{1}{c_\tau^2} \cdot \mathrm{L}_\tau(x, y_\tau, y'_\tau; w_\tau) + \ln(1 + c_\tau^2). \tag{3.5}$$

As we demonstrate in Chapter 5, the use of AWL allowed the method to improve its results for both the specialization step as well as the aggregation step. Thus, we decided to include it in our final proposed method, FGFR+.

3.4 Self-Attention

During the development of FGFR+ we also experimented with different ways of combining the outputs generated by the heads of the model. We hypothesized that self-attention could be an appropriate combination method that, ideally, would identify which heads are providing more relevant information for the image at hand (somehow acting as a 'soft subset selector' [17]). In this section, we briefly describe the popular attention methods introduced in [59], ranging from the scaled dot product attention mechanism to the transformer encoder architecture. We opted to use these methods given the recent success of attention-based methods in a wide variety of use cases.

The authors in [59] introduce a generic definition for what an attention function is: a mechanism that maps a query and a set of key-value pairs to an output, where the queries, keys, values and output are all vectors. The output is simply computed as a weighted sum of the values. The weight of each value corresponds to the result of a compatibility function between the query and the corresponding key.

In their work [59], they introduce a novel compatibility function, 'scaled dot-product attention'. It first computes the dot product between the query and all the keys, then divides the resulting values by a constant $\sqrt{d_k}$ and applies a softmax function to obtain the weight values. The output is computed as the weighted sum of the values. Both the keys and queries are of dimension d_k, whereas the values have a dimension of d_v.

To compute all the outputs simultaneously, all the queries, keys and values are packed together into matrices Q, K and V. The following formula is used to compute the matrix of outputs:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^{\text{T}}}{\sqrt{d_k}})V \tag{3.6}$$

This mechanism accepts an N-sized sequence of vectors, and generates another N-sized sequence of vectors, where there is a one-to-one correspondence between the vectors of the input and the output. An output vector can be understood as a modification of its corresponding input vector that incorporates relevant information extracted from other elements in the sequence.

Building on top of the scaled dot-product attention, the authors in [59] introduce the *multi-head attention module*, which performs h parallel scaled-dot product attention mechanisms for a single sequence. Assuming that the input sequence consists of vectors of dimension d_{model}, each of the vectors will be projected into a d_k dimensional space, where $d_k = d_v = d_{\text{model}}/h$. This allows each parallel attention layer or head to attend to information from different representation subspaces. For each input vector, each of its corresponding h output vectors of dimension d_v will be concatenated and projected using a feed-forward layer, retaining the initial d_{model} dimensionality. This process is illustrated in Figure 3.6.

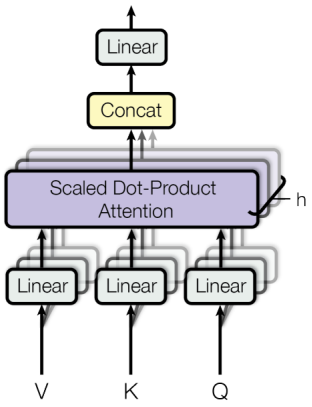


Figure 3.6: Diagram showing the structure of the multi-head attention module [59]. Figure borrowed from [59].

Finally, in this work we also experimented with transformer encoders [59]. A transformer encoder layer is composed of two sub-layers: a multi-head attention module, and a simple, position-wise fully connected feed-forward layer. Each of the sub-layers is surrounded by a corresponding residual connection [23], followed by layer normalization [1]. This structure is shown in Figure 3.7.

Even though the experimentation with self-attention constituted an important part of our work, we decided not to include it in our final method proposal, given that our empirical results clearly indicated that its inclusion would lead to worse performances compared to other methods we tested for the combination of head outputs, as we describe in more detail in Chapter 5.

3.5 Focal Loss

We also explored modifying the single-task loss function used by FGFR. Cross-entropy is the single-task loss function used throughout the three steps of the method, including the training of the specialized heads (the cross-entropy loss of each head is computed and then aggregated

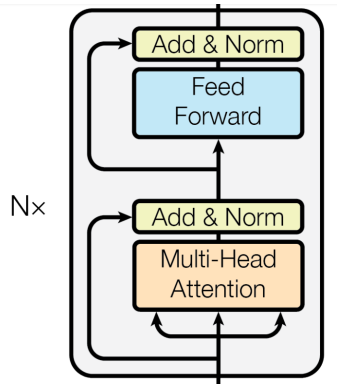


Figure 3.7: Overview of the architecture of the transformer encoder [59]. Figure borrowed and adapted from [59].

to obtain a single, aggregated loss value), as well as the training of the baseline model and the aggregation step. We argue that the method might benefit from using other kinds of loss functions, such as the popular focal loss.

Focal loss [37] was introduced as a method to address class imbalance by modifying the standard cross-entropy so that the easily classified examples are assigned a smaller fraction of the loss. Its main difference with balanced cross-entropy is the fact that focal loss down-weights the loss assigned to easy examples, forcing the model to focus on hard negatives. Standard cross-entropy is defined as follows:

$$\mathrm{CE}(p_t) = -\log(p_t), \tag{3.7}$$

where p_t is the probability distribution for the current prediction. Focal loss includes a modulating factor $(1 - p_t)^\gamma$ to cross-entropy with a tunable focusing parameter $\gamma \geq 0$. Focal loss can then be defined as follows:

$$\mathrm{FL}(p_t) = -(1 - p_t)^\gamma \log(p_t) \tag{3.8}$$

Note how, whenever an example is misclassified with a low confidence, the modulating factor will be near 1 (as p_t will be close to 0), which results in the loss being almost unaffected. Whenever p_t is close to one, the modulating factor becomes closer to zero, weighting down the loss assigned to well-classified examples. The γ parameter determines the rate at which easy examples are down-weighted. Note that, whenever $\gamma = 0$, focal loss is equivalent to cross-entropy. Also note how the effect of the modulating factor grows as the value for γ is increased.

Focal loss is typically used in conjunction with α balancing, so that each example is weighted according to the inverse frequency of its class. The complete formulation for α-balanced-focal loss would be the following:

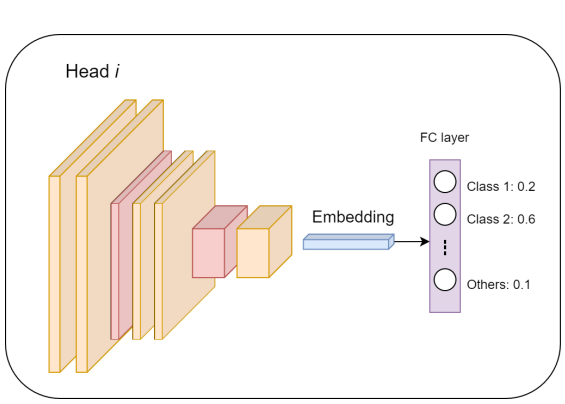
$$\mathrm{FL}(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t). \tag{3.9}$$

It must be noted, however, that even though we performed experiments with this loss function, we eventually decided to not include it in our proposed FGFR+ (as we explain in more detail in Chapter 5), both for our limited computation and temporal resources, as well as the little improvements we observed from its use. Consequently, we decided to keep using cross-entropy as the main loss function for FGFR+, also relying on AWL for the aggregation of the head losses during the specialization step.

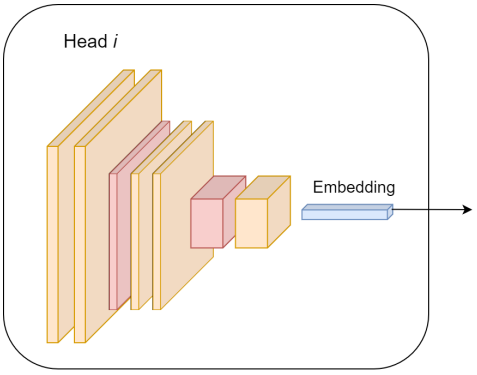
Chapter 4

Proposed Method: FGFR+

We propose two main improvements over the original FGFR [51], based on the potential weaknesses we identified during our study of the original method described in detail in Section 5.2. Our modifications are conceptually simple yet effective, as shown in the results reported in Section 5.3. We introduce separate modifications for both the specialization and the aggregation steps.



(a) Specialization step: the embedding is generated and a prediction is made on top of it using a FC layer within the head.



(b) Modified aggregation step: the output of the head is the embedding, and the FC layer of the head is removed (this is our proposed approach).

Figure 4.1: Diagram illustrating the internal structure of a head and how it varies between the specialization and aggregation steps in our new approach. In the original aggregation step, the FC layer within the head was maintained, outputting prediction logits rather than representations. Note that, even though the diagrams show that the embedding generation method are convolutional layers, this is only done for illustration purposes, and any other architecture such as Transformers could be used instead.

4.1 The FGFR+ Method

We now describe each of the modifications we introduce to extend the original FGFR, for both the aggregation step and the specialization step. We did not introduce any modifications for the initial step, as we did not consider neither the training of the baseline nor the multi-subsets extraction to be one of the major bottlenecks of the original method.

4.1.1 Our Proposal for the Aggregation Step

Most of our work was focused on improving the aggregation step, as we believed that the aggregation method used in the original FGFR was not taking full advantage of the potential of the multi-head model. We found the training of the original aggregation step to be difficult and quite unstable. We hypothesized that the main bottleneck of this step was the fact that all the information generated by each of the heads was being passed to the final FC layer as mere prediction logits (i.e., the output of the FC layer at the end of each of the heads). More concretely, we find that, while the FC layers of the individual heads are necessary to train the specialization step (Figure 4.1a), removing these per-head classifiers during the aggregation step allows to fully extract the potential of the method. As a result, the output of each of the heads will no longer be prediction logits, rather than image embeddings (what used to be the input to the per-head classifiers), which we believe contain much richer information, allowing a more efficient flow of information between the heads and the final aggregation layer. This is illustrated in Figure 4.1, where it can be seen that, during the specialization step, the heads output prediction logits generated by their classifier FC layers, whereas, during the new aggregation step, the heads directly output embeddings, without making use of the per-head classifiers.

As mentioned above, our proposal starts by removing the per-head FC classification layer of each of the heads (both the specialized heads and the baseline head), so that each of them outputs an image embedding (a D dimensional vector representation of the image, where D will vary depending on the baseline model architecture used). Consequently, for each input image passed to the network, there will be as many embeddings as there are heads. We propose combining these embeddings using their element-wise mean, so that the resulting combined embedding retains the same dimensionality as the single-head embeddings. The resulting combined embedding will then be fed to a final FC classification layer, which will be in charge of making the final prediction. This whole process is illustrated in Figure 4.2, where the head embeddings are combined using their element-wise mean (maintaining the original embedding dimensionality), and the resulting embedding is passed to the final classification layer, whose expected input dimensionality is set to match the shape of the head embeddings.

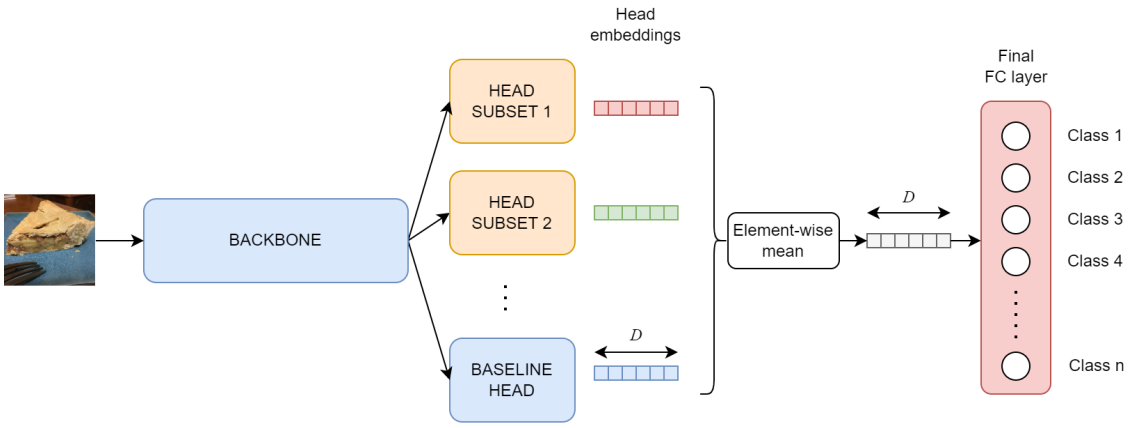


Figure 4.2: Diagram illustrating the proposed aggregation method.

Additionally, we propose training the weights of the complete model during this step, instead of solely training the final FC layer. We find that this approach allows to unlock the full potential

of the model, giving the heads and the backbone flexibility to optimize their weights for the aggregation step. It is worth mentioning that the original aggregation step did not benefit from unfreezing all the model weights, unlike ours, which shows a significant improvement (see Section 5.4.1 for more details).

4.1.2 Our Proposal for the Specialization Step

We also propose a modified specialization step. This is the step where the actual fine-grained learning takes place, and is the cornerstone of the whole FGFR method. The original specialization step was trained to optimize an aggregated loss function that was defined as a simple average of the individual cross-entropy losses of the specialized heads. After consulting the multi-task learning literature, we found that there are other existing ways of combining single-task losses in a more effective way than a mere average. Works such as [32] and [36] introduce single-task loss weighting methods that are simple to implement and show promising results. We found that the specialization step from FGFR can also benefit from these approaches.

Our improvement consists of making use of the loss weighting method introduced in [36] (which we refer to as AWL, Automatic Weighted Loss) to perform the multi-task training, so that the final loss function is formulated as a weighted sum of the cross-entropy losses of each of the specialized heads. As a result, the weight for the loss of each task is a parameter that is learned jointly with the weights of the model, which allowed us to improve the performance in most of the cases without increasing the complexity of the model.

The final formulation of the loss for our proposed specialization step that makes use of AWL is then defined as follows [36]:

$$\mathcal{L}_{\mathrm{T}}(x, y_{\mathrm{T}}, y'_{\mathrm{T}}; w_{\mathrm{T}}) = \sum_{\tau \in \mathrm{T}} \frac{1}{c_\tau^2} \cdot \mathrm{L}_\tau(x, y_\tau, y'_\tau; w_\tau) + \ln(1 + c_\tau^2), \tag{4.1}$$

where L_τ represents the cross-entropy loss of head τ, and c_τ represents the learned parameters that determine the weight of each of the losses.

The rest of the aspects of the specialized step are maintained from the original method: only the specialized heads are trained during this step, and the original data balancing procedures are still used in the new approach.

Chapter 5

Evaluation

In this chapter, we present every aspect related to the evaluation we performed of both the original FGFR and FGFR+ methods. We begin by describing in Section 5.1 the details about the datasets and the experimental setup we used. In Section 5.2, we present a summary of our experimentation process with the original FGFR, showing its ability to generalize to a variety of domains and backbone architectures, as well as providing some insights into its behavior. In Section 5.3, we present the results obtained with FGFR+ and compare them to both the original method and some other state-of-the-art approaches for fine-grained image recognition. Finally, in Section 5.4, we report our analysis of the most relevant sets of experiments we performed during the development of FGFR+, evaluating the impact of a variety of modifications that we applied on top of the original approach.

5.1 Datasets and Experimental Setup

In this section, we introduce the datasets used for the evaluation of the original FGFR and FGFR+, as well as the complete experimentation framework and settings we employed.

5.1.1 Datasets

In their work [51], the authors of the original FGFR only evaluated their method on food image datasets: Food-101 [6], FoodX-251 [31] and Food1K (a sub-dataset built from Food2K [46]). In order to assess the generalization capability of the method, we perform experiments using three datasets of notably different domains and sizes: CUB-200-2011 [60] (birds), Stanford Cars [34] (cars) and Food-101 [6] (food dishes). The main reason why we also experiment with the Food-101 dataset is to be able to confirm the results reported in [51].

CUB-200-2011 is one of the most commonly used datasets for evaluating fine-grained visual analysis tasks. It consists of more than 11,000 images of 200 different bird species, split into two train and test sets of 5,994 and 5,794 images, respectively. Even though each image contains multiple annotations (e.g., class, part locations and bounding boxes), we only use the class annotations for the task of multi-class classification. We include some examples of images from this dataset

in Figure 5.1. The training set is virtually balanced, with all the classes containing 30 instances except for 6 which consist of 29 images. The test set is slightly more imbalanced, as seen in Figure 5.2, with five classes containing less than 20 images.

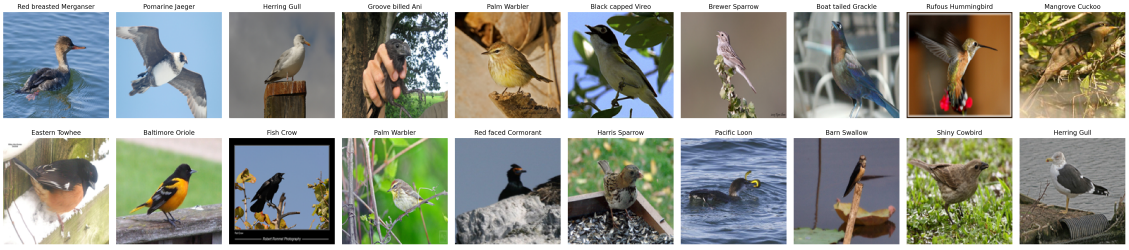


Figure 5.1: Sample images from the CUB-200-2011 dataset.

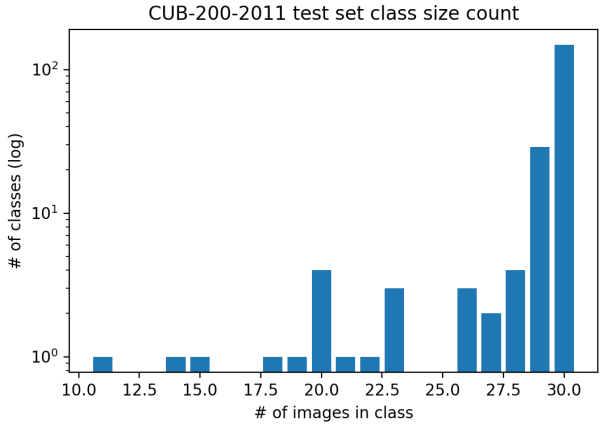


Figure 5.2: Class size count for the CUB-200-2011 test set.

Stanford Cars is another popular fine-grained dataset consisting of images of a variety of real cars. The dataset contains 16,185 images in total, evenly split into train and test sets of around 8,000 images each. The dataset consists of 196 classes, each class typically being associated with a particular combination of car brand, model and year (e.g., 2012 BMW M3 coupe). Example images from this dataset are shown in Figure 5.3. Given that on this project we focus on multi-class classification, we use the cropped version of the dataset, cropping each image according to its associated bounding box information, so as to make the classification based solely on the main car shown in each image. Both the train and test sets present significant class imbalance, as shown in Figure 5.4.

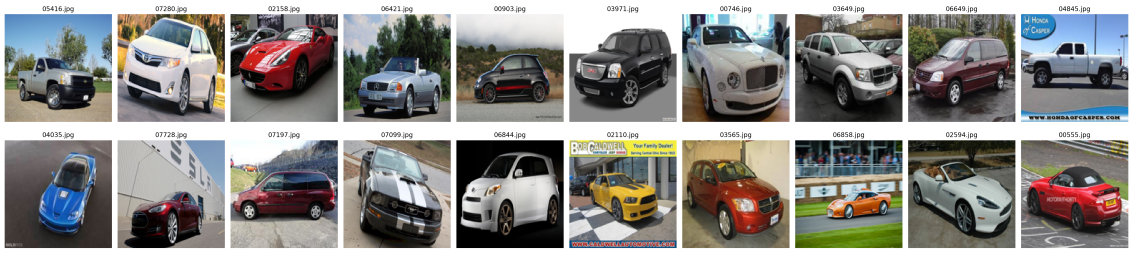


Figure 5.3: Sample images from the Stanford Cars dataset.

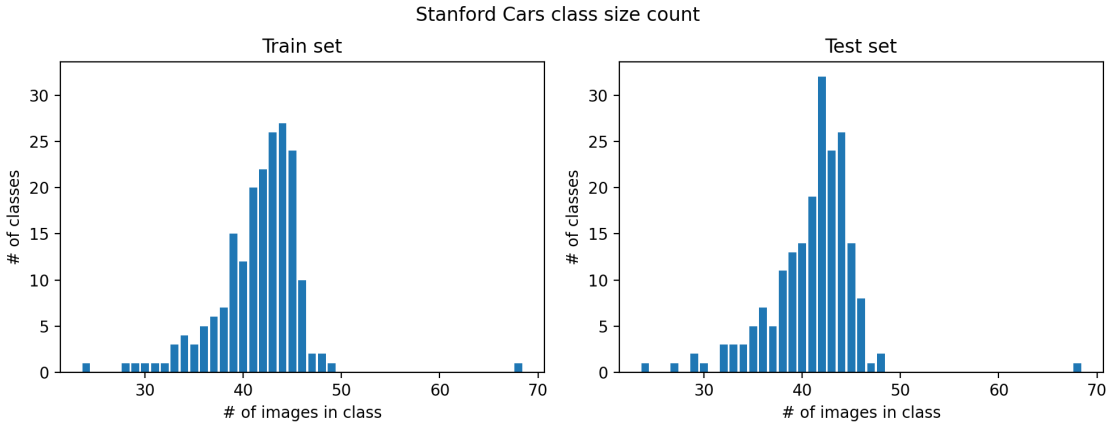


Figure 5.4: Class size counts for the Stanford Cars train and test sets.

Food-101 is a popular dataset consisting of 101,000 images of food belonging to one of 101 food categories. The dataset is, in principle, perfectly balanced, with 750 and 250 images per class for the train and test sets, respectively. However, as the authors state [6], the dataset presents some annotation issues, namely the problem of labeling an image of multiple food items with a single class, resulting in some cross-category noise. Sample images from this dataset can be found in Figure 5.5.

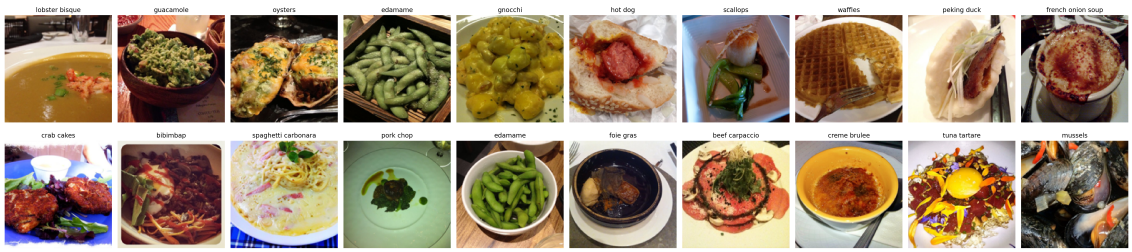


Figure 5.5: Sample images from the Food-101 dataset.

5.1.2 Framework and Tools

We made use of the Python programming language for the development of the project, relying on the PyTorch framework [47]. We mostly used the model implementations provided by the `torchvision` [44] package, except for the CSWin model [13], for which we directly used the implementation provided by its authors[1]. We made extensive use of the `timm` library [63] to leverage the many utilities it implements. We used Weights & Biases [5] to track and visualize our experiments. The experimentation process was carried out using a NVIDIA GeForce RTX 3090 GPU.

5.1.3 Baseline Models

Two different baseline architectures were used for the experimentation process: EfficientNet [55] and CSWin Transformer [13].

[1]Implementation available in https://github.com/microsoft/CSWin-Transformer

EfficientNet is one of the most widely used CNN architectures. Thanks to its scaling method that uniformly scales all dimensions in model depth, width and resolution, there are multiple EfficientNets of varying sizes, ranging from the EfficientNet-B0 (5.3M parameters) to the EfficientNet-B7 (66M parameters). For this project, we opted to use the smallest variant, B0, for a number of reasons.

First, to confirm that the original FGFR proposed in [51] is also applicable to relatively small models (given that they only reported results using large models, with $\approx$ 173M and $\approx$ 296M parameters). Second, a small model would allow us to significantly speed up the experimentation process, while also drastically reducing the amount of computational resources required. The choice for the EfficientNet architecture was taken carefully: we wanted to evaluate the performance of the method on convolutional models (the original work [51] only reported results for attention-based models, CSWin-Transfomer [13] and VOLO [64]). Within the field of CNNs, we chose the EfficientNet family given not only its popularity, but also its strong performance over methods with a similar amount of parameters (e.g., ResNet [23] or DenseNet [27]).

To integrate this model with the FGFR framework described in [51], a separation point within the model had to be determined, so that the model backbone and head can be separated in order to perform the specialization and aggregation steps of the method. Given that the main building blocks of the EfficientNet are seven mobile inverted bottleneck MBConv blocks [53, 56], we opted to make the separation after the sixth block, so that the head consists of the seventh block and the final layers after it (a 1x1 convolution, a pooling layer and a fully connected layer). Figure 5.6 shows a diagram visualizing this split.

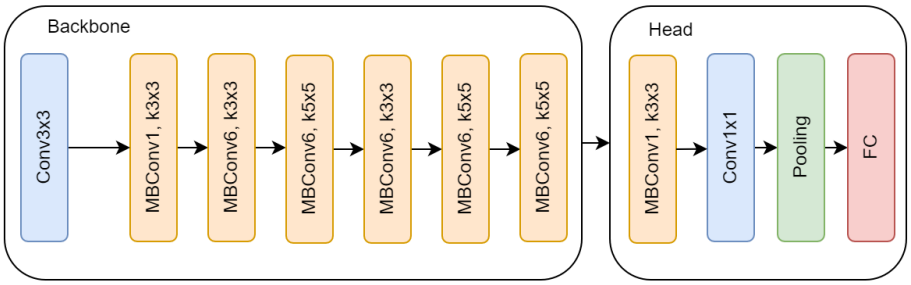


Figure 5.6: Diagram showing the selected split between backbone and head for the EfficientNet architecture.

CSWin Transformer is a Transformer-based backbone that can be used for a variety of vision tasks. The authors published multiple variants, ranging from 23M parameters (CSWin-T) to 173M (CSWin-L). For this project, we opted to use the CSWin-L model, as it was also used to evaluate the original method in [51]. We use CSWin-L as the counterpart of EfficientNet-B0: a large, attention-based model. In this way, we can compare results obtained by two very different baseline models.

Regarding the split between backbone and head, we followed the same approach as [51]: out of the four CSWin Transformer blocks, the first three are considered part of the backbone, and the last one is used as the head.

5.1.4 Training Configuration

All models were trained using 224x224 RGB input images, following [51]. Prior to the training of the baseline, both the EfficentNet and CSWin models were initialized using the pretrained weights for ImageNet [52]. The batch size varied depending on the model used, as the sizes of the two baseline models vary significantly. The batch size would also be affected by the FGFR step that was being trained, as different steps would require different amounts of memory. For instance, in the case in which the number of generated subsets is high, the training of the specialization step would require significantly more GPU memory than the initial step, as there would be many replicated heads to be trained. Thus, the batch size for the specialization step would need to be smaller. The batch size used for the first step (baseline training) was 256 for the EfficientNet-B0, and 32 for CSWin-Large. However, from this point on, the batch size used for each dataset was adapted (if needed) to maximize the amount of memory used without exceeding its limit, depending on the number of generated clusters. At some points in this document we report results obtained using gradient accumulation (GA) [26], to effectively increase the batch size without requiring additional memory. We only use GA in those cases where we explicitly state it.

The optimizer used throughout the development of the project was AdamW [41] with betas of 0.9 and 0.999, and an epsilon value of 1e−8. A weight decay of 0.005 was used in all experiments. Regarding the learning rate (LR) scheduler, a cosine annealing scheduler with warm restarts [42] was used. The initial T_0 was set to 10, with a multiplier of 1.2 after each restart, and a total of 4 restarts. After each restart, the initial learning rate was multiplied by a factor of 0.75 unless otherwise stated. In total, this configuration led to 63 epochs of training.

Regarding the learning rate, we found it to be sensitive to three factors: the baseline model (EfficientNet or CSWin), the dataset and the step within the method (the baseline training, the specialization step or the aggregation step). Consequently, for each of these scenarios, the learning rate was tuned using a grid search over different learning rates (typically ranging from 1e−6 to 1e−4). Whenever the multi-task loss weighting methods from [32] or [36] were used, a separate identical AdamW optimizer was used with a constant learning rate of $1e-3$.

The selected loss function for training was cross entropy, for each of the three steps of the pipeline. In the particular case of the specialization step, each of the heads is trained using its cross entropy loss, and then the final loss value of the whole model is computed by combining all the individual head losses using either their mean (original FGFR) or Automatic Weighted Loss [36] (FGFR+).

The main evaluation metric was the top-1 validation accuracy across the three steps (following [51]). However, specially during the specialization and aggregation steps, a more detailed quantitative evaluation was performed by manually analyzing per-class metrics (e.g., precision, recall and F1) for each of the classes, in order to assess what the model was learning. During the training of the specialization step, given that each head would have its own top-1 accuracy, the overall evaluation metric was the average balanced top-1 accuracy across all heads.

Additionally, for some of our experiments we report the overall top-1 validation accuracy together with two other accuracy metrics: the generic accuracy and the cluster accuracy. The former refers to the top-1 accuracy obtained for the subset of the validation set corresponding to images that do not belong to any cluster. The latter corresponds to the top-1 accuracy for the subset of the

validation set corresponding to images that belong to a cluster.

5.1.5 Data Augmentation

As stated in Section 5.1.4, all models were fed with 224x224 RGB images. Even though we tried to keep the image preprocessing pipeline consistent across all datasets and models, we found that, for the particular case of the Stanford Cars dataset, better results were consistently obtained when using a simpler augmentation pipeline.

The data augmentation operations applied when training for the CUB-200-2011 and Food-101 datasets are the following:

- Random resized crop and interpolation (cropping a random size and aspect ratio with random interpolation) of size 224x224, random scale between 0.5 and 1.0, and random aspect ratio between 0.75 and 1.33.
- Random horizontal flip with a probability of 0.5.
- AugMix augmentations [25], with $\alpha = 1.0$, width $= 3$ and a random depth value between 1 and 3.
- Channel-wise normalization using ImageNet means and standard deviations.
- Random Erasing [68] of a rectangular area of the image, with a probability of 0.25, a minimum erased area with respect to the image size of 0.02, and a maximum area fraction of 0.33.

We found that omitting both the AugMix and Random Erasing augmentations consistently led to better results when training for the Stanford Cars dataset. Consequently, for this dataset, we replaced these strong augmentations with a simple random rotation within the range $(-15°, 15°)$.

The preprocessing pipeline for validation images was the same for all datasets, and can be described as follows:

- Image resizing to 248x248.
- Center cropping of size 224x224.
- Channel-wise normalization using ImageNet means and standard deviations.

5.2 Exploring the Original FGFR

We performed an in-depth study of the behavior of the original FGFR as it was described in [51]. We focused on analyzing each of the most relevant aspects of the three main steps of the pipeline: the multi-subset extraction, the specialization step and the aggregation step.

5.2.1 Study of the Classes Subset Extraction

In this section, we assess the generalization ability of the classes subset extraction method to different domains and baseline model architectures. We qualitatively confirm that the classes

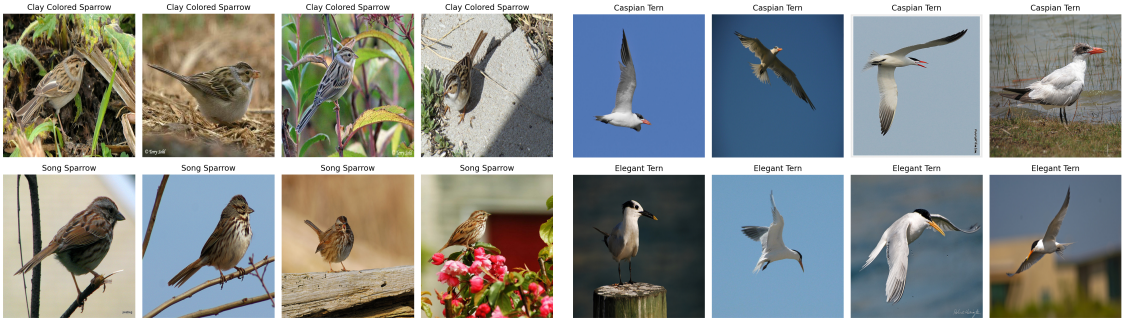
within a cluster generally present small inter-class variance, making them 'hard' to distinguish. We also corroborate the findings reported in [51], showing that the obtained clusters tend to be also present in the embedding space of the model.

We start by performing the initial step of the original FGFR, the training of baseline, for all six different combinations of datasets and models. Each dataset-model configuration is carefully trained by performing a grid search over the most relevant hyperparameters (namely the learning rate and the set of data augmentations used). Table 5.1 shows the top-1 validation accuracy of the best model obtained in each scenario.

Top-1 validation acc.	**EfficientNet-B0**	**CSWin-Large**
CUB-200-2011	79.44	90.21
Stanford Cars	91.78	93.76
Food-101	87.56	94.10

Table 5.1: Best top-1 validation accuracy for each dataset-model pair, for the initial step of the method (baseline training).

After applying the class subset extraction procedure introduced in [51] and explained in Section 3.1.1, we qualitatively evaluate the similarity of classes belonging to the same cluster. Figures 5.7, 5.8 and 5.9 show some examples of images of classes belonging to several clusters, for different datasets.



(a) CUB-200-2011, CSWin-Large, cluster 1.

(b) CUB-200-2011, CSWin-Large, cluster 2.

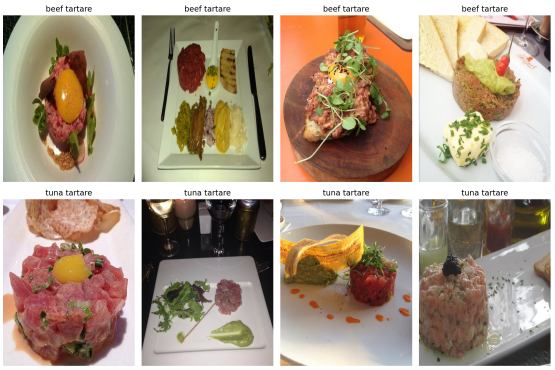
Figure 5.7: Images sampled from two clusters generated for the CUB-200-2011 dataset using CSWin-Large as a baseline. Each row contains images belonging to the same class.

(a) Stanford Cars, EfficientNet-B0, cluster 1.

(b) Stanford Cars, EfficientNet-B0, cluster 2.

Figure 5.8: Images sampled from two clusters generated for the Stanford Cars dataset using EfficientNet-B0 as a baseline. Each row contains images belonging to the same class.

It can be seen how, regardless of the dataset, the subset extraction method is capable of finding



(a) Food-101, CSWin-Large, cluster 1.

(b) Food-101, CSWin-Large, cluster 2.

Figure 5.9: Images sampled from two clusters generated for the Food-101 dataset using CSWin-Large as a baseline. Each row contains images belonging to the same class.

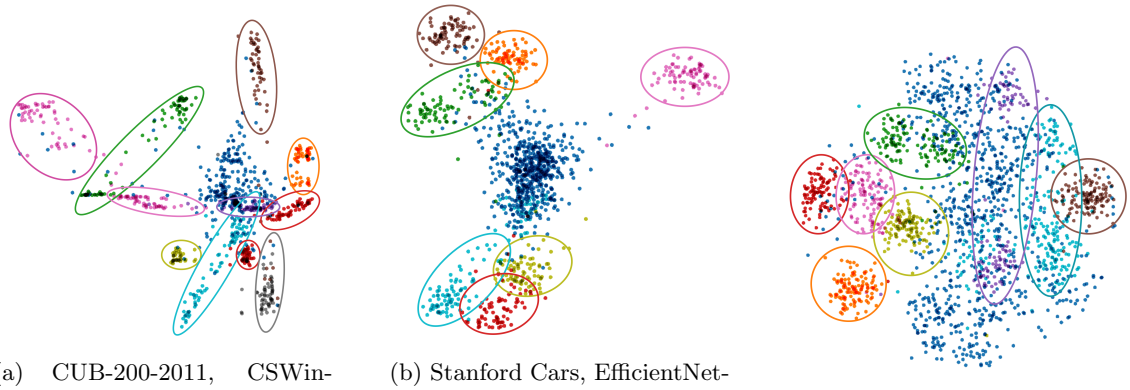
clusters of classes that are visually similar. Moreover, it is clear how the generated clusters tend to align with different real life object categories, such as sparrows or terns for the CUB dataset (Figure 5.7), 2012 Audi cars or 2009 Bugatti Veryons for Stanford Cars (Figure 5.8), or tartars or different kinds of cooked meat for Food-101 (Figure 5.9).

We also evaluate the correlation between the obtained clusters and the embeddings learned by the model, to verify that classes within the same cluster are hard to distinguish not only for humans, but also for the models. To that end, we record the image embeddings or representations learned by each of the models (i.e., the N-dimensional vector that would be fed to the final classification layer) and visualize them in two dimensions using t-SNE [58]. Figure 5.10 shows the 2D embedding visualizations for three different dataset-model pairs, also including the location of the different clusters generated using the subset extraction procedure.

We find that images within the same cluster tend to have similar embedding representations. The visual analysis also shows that there is certain overlap between some clusters in the embedding space, suggesting that, in some cases, distinguishing between two different clusters could also be difficult (for instance, when two clusters represent sub-categories of a larger category).

5.2.2 Study of the Specialization Step

Following the generation of the clusters, the next step of the method involves the creation of the multi-subset classifier and the training of the specialized heads. This part of the document aims at providing an insight into what the specialized heads are capable of learning during this step. To that end, we build on top of the trained baselines mentioned in Section 5.2.1, with their corresponding clusters, and perform the specialization step for each of them. Again, we carefully train each of these models performing a grid search over the most relevant hyperparameters, namely



(a) CUB-200-2011, CSWin-Large.

(b) Stanford Cars, EfficientNet-B0.

(c) Food-101, CSWin-Large.

Figure 5.10: t-SNE visualization of the embeddings generated by different baselines for all datasets. Each point represents an image. The color of a point represents the cluster it belongs to. For the sake of clarity, an ellipse has been placed around points belonging to the same cluster.

the learning rate.

Following the procedure described in [51], for each dataset-model pair we replicate the head of the baseline as many times as there are clusters, adapting the final classifier of each head to the number of classes belonging to the cluster plus one, to account for the 'others' class. To train the multi-task model, the cross-entropy loss of each head is calculated and averaged to obtain a final loss for the complete model. During this step, neither the shared backbone part of the model nor the baseline head are trained, as they were already trained in the previous step. We follow the data balancing method described in [51].

In order to analyze the learning of the specialized heads, we first inspect the evolution of the balanced top-1 validation accuracy of each of the heads. The use of the *balanced* accuracy is important, as each specialized head presents strong class imbalance between the class 'others' and the rest of the classes in the cluster. We include an example of the evolution of this metric for the case of Stanford Cars with the EfficientNet architecture in Figure 5.11. The balanced top-1 accuracy metric is computed individually for each head, measuring its classification accuracy for the classes within its cluster and for its corresponding 'others' class. We also report the average balanced top-1 validation accuracy across all heads, in order to be able to evaluate the learning of the model with a single metric.

Figure 5.11 shows that, in general, all heads seem to be improving their performance throughout the training process. However, we notice a problem that is common within the field of multi-task learning: different tasks (heads) have different learning speeds. Consequently, some heads will achieve their peak performance early on during the training process, while others will peak during the last epochs. This introduces some problems, such as the difficulty of finding an appropriate LR schedule that suits all tasks.

Once confirmed that the performance of the specialized heads is indeed improving during training, we compare, class by class, the performance of the specialized heads against the performance of the general baseline trained during the initial step of the method. We only consider those classes that are present in the clusters (i.e., the 'hard' classes). We measure the per-class precision and

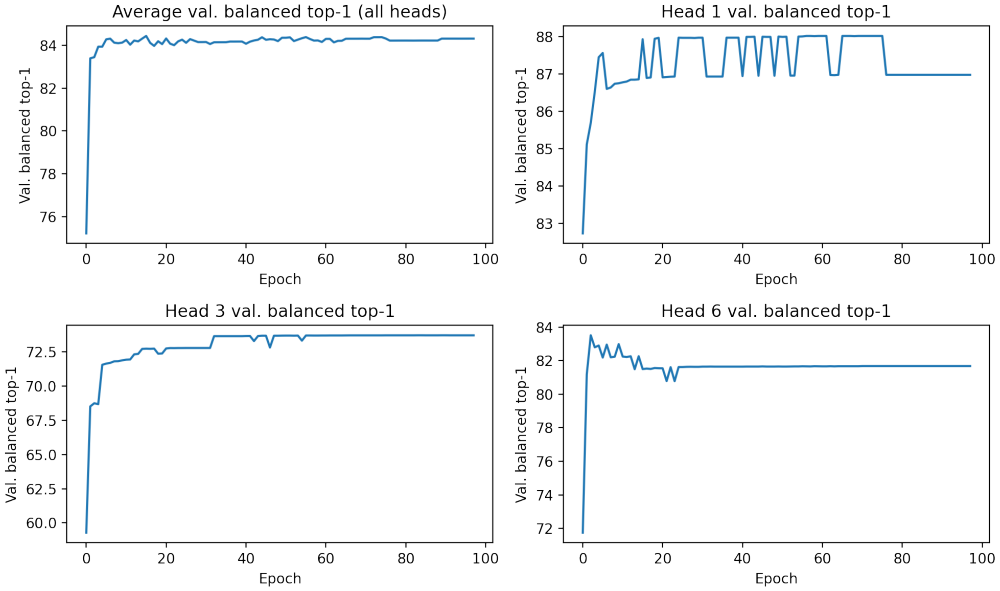


Figure 5.11: Some example balanced top-1 validation metrics for the training of the specialization step (Stanford Cars + EfficientNet-B0).

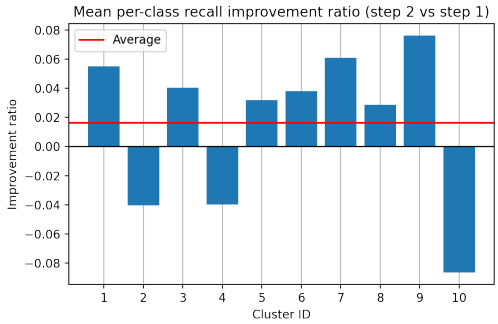
recall for both the initial baseline (with a single confusion matrix covering all classes) and for the specialized heads (where each head will have its own confusion matrix for its corresponding subset of classes, including the 'others' class). Table 5.2 shows the per-class precision and recall metrics for the model after the training of the baseline (step 1) and the specialization step (step 2), for the particular case of Stanford Cars with EfficientNet (validation set).

An important observation to make from Table 5.2 is that, in general, the per-class recall for the cluster classes increases during the specialization step, hinting that the model is improving its ability to correctly identify instances of the 'hard' classes (visualized in Figure 5.12). However, the precision is typically lower in the specialization step. After a thorough analysis, we found that the precision metric for each of the heads is extremely sensitive to the classification performance of the 'others' class. When the recall for the 'others' class is low, there will be some 'others' images incorrectly classified as 'subset' classes. Considering the strong class imbalance between the class 'others' and the rest, even a seemingly high recall of 0.99 for 'others' can lead to a significant deterioration in the precision of the rest of the classes in the subset. Figure 5.13 illustrates this problem.

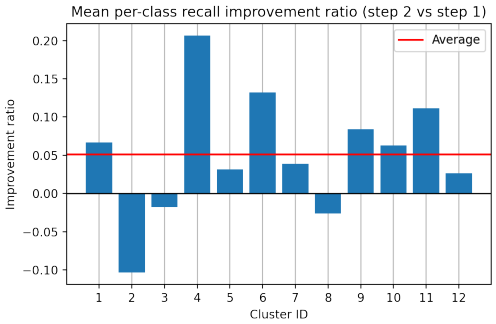
We found a significant difference between EfficientNet and CSWin models in terms of the classification performance for the 'others' class, as CSWin would typically lead to superior results for this class. Consequently, the above mentioned issue is not as accentuated as it is for the EfficientNet models. However, we hypothesize that this problem, wrongly classifying 'generic' classes as 'hard' classes, is partially mitigated during the final step, the aggregation step, as the model will have at its disposal the information generated by the rest of the heads (including the original baseline head), and thus might be able to *ignore* the mistaken prediction made by this head by focusing more on other heads with more confident estimations. In any case, the improvement of the per-class recall is consistent across all dataset-model pairs (see Figure 5.12 for some examples),

Cluster ID	Class	Step 1 Precision	Step 1 Recall	Step 2 Precision	Step 2 Recall
1	others	-	-	0.999	0.996
1	44	0.815	0.688	0.568	0.781
1	45	0.776	0.884	0.661	0.860
2	others	-	-	1.000	0.993
2	63	0.538	0.483	0.364	0.552
2	70	0.513	0.571	0.429	0.600
2	118	0.824	0.824	0.559	0.765
3	others	-	-	0.999	0.989
3	68	0.767	0.868	0.376	0.842
3	74	0.865	0.727	0.443	0.795
4	others	-	-	0.999	0.994
4	15	0.769	0.698	0.560	0.651
4	16	0.651	0.700	0.416	0.800
5	others	-	-	1.000	0.985
5	12	0.717	0.927	0.411	0.902
5	21	0.844	0.643	0.274	0.738
6	others	-	-	1.000	0.985
6	41	0.774	0.706	0.273	0.706
6	42	0.761	0.761	0.343	0.804
7	others	-	-	1.000	0.989
7	13	0.706	0.571	0.400	0.619
7	18	0.580	0.725	0.304	0.775

Table 5.2: Per-class precision and recall of cluster classes between the model after step 1 (baseline training) and step 2 (specialized heads training), for the Stanford Cars dataset and using an EfficientNet model. Only a subset of the clusters is shown for the sake of readability.



(a) Stanford Cars, EfficientNet-B0.



(b) CUB-200-2011, CSWin-Large.

Figure 5.12: Diagrams showing the mean per-class improvement of the recall metric for each cluster, in two different scenarios.

potentially leaving room for the aggregation step of the method to improve the final classification with respect to the baseline model.

5.2.3 Study of the Aggregation Step

After training the specialized heads, the last part of the method consists in aggregating the outputs of each of the heads to make the final prediction. To train this final step, the procedure described in [51] is replicated: the output logits from the FC layers of each head (the specialized heads and

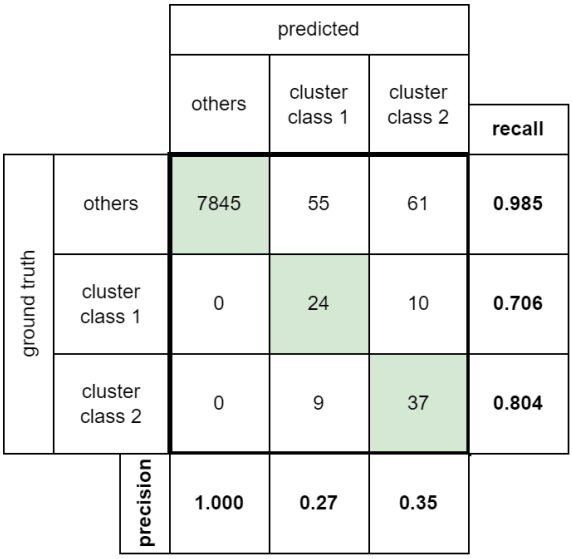


Figure 5.13: Example of a specialized head confusion matrix. Despite the 'others' class having a seemingly high recall of 0.985, it is low enough for it to significantly affect the precision of the rest of the classes.

the original head) are concatenated and passed as input to a final classification FC softmax layer, which will be in charge of making the definitive prediction. During this step, the whole model is kept frozen, only training the final FC layer. This layer is initialized using the identity matrix, so that the model retains at least the performance of the baseline.

For each dataset-model pair, we take the best model trained during the specialization step, based on the balanced top-1 validation accuracy, and we train the aggregation step on top of it. We carefully train each model with different learning rates, and report the results in Table 5.3. In order to evaluate the real contribution of the model, we report the validation results without Test Time Augmentation (TTA) [33].

Dataset	**Model**	**Baseline Val. Top-1**	**Final Val. Top-1**	**Gain**
CUB-200-2011	EfficientNet-B0	79.44	**79.51**	0.07
	CSWin-Large	90.21	**90.40**	0.19
Stanford Cars	EfficientNet-B0	91.78	**91.89**	0.11
	CSWin-Large	93.76	**93.84**	0.08
Food-101	EfficientNet-B0	87.56	**87.68**	0.12
	CSWin-Large	94.10	94.10	0.00

Table 5.3: Results obtained with the complete method for all dataset-model pairs (without TTA).

The results from Table 5.3 show that the method is capable of improving with respect to the performance of the baseline, as it is doing so in five out of six scenarios. However, in some cases, the improvement is practically insignificant, with top-1 accuracy gains of just a few hundredths. We compare these gains with those originally reported in [51] with the original FGFR under the same settings, without TTA. Despite not sharing all the datasets and models they use, we find our results to be arguably similar to theirs, with gains of around one or two decimal points, with a few exceptions.

Even though the results show that the method is capable of improving upon the classification performance of the baseline model, we found the aggregation step to be quite difficult to train. The results reported in Table 5.3 were obtained after an exhaustive search over different values for the learning rate and some other hyperparameters of the LR scheduler. We found this step to be very sensitive to the learning rate used. Whenever this value was too big, the initial accuracy of the model (that of the baseline, thanks to the identity matrix initialization) would quickly drop without ever recovering. However, decreasing the LR would make the weights of the final FC layer remain almost unchanged, leading the final model to almost replicate the behavior of the baseline (this is the case for Food-101 + CSWin-Large, in Table 5.3). This led to each configuration having a narrow range of LR values that allowed for proper learning. And, even in those cases where we managed to improve with respect to the baseline, the curve of the accuracy metric (Figure 5.14) shows that the learning is quite unstable.

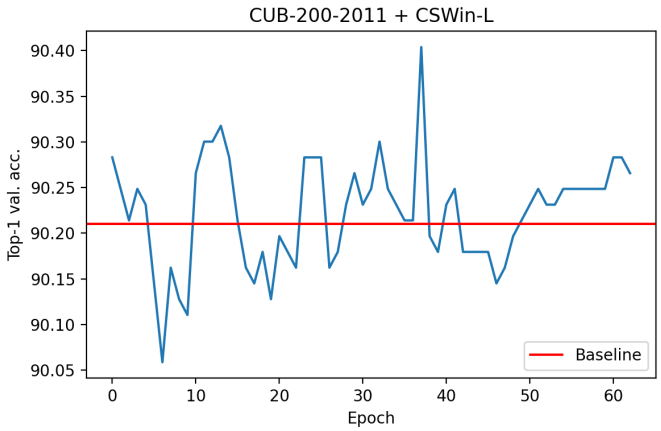


Figure 5.14: Evolution of the top-1 validation accuracy during the aggregation step for CUB-200-2011 + CSWin-Large.

Following [51], we also experimented with different values for gradient accumulation (GA) [26], to check whether increasing the effective batch size could result in more stable learning. We found that using gradient accumulation for this step did not have a remarkable effect in either the accuracy or the learning stability. The results in Table 5.3 were obtained using a gradient accumulation value of 10 (i.e., increasing the effective batch size tenfold). Removing GA would lead to slightly worse accuracies (e.g., 79.44 versus 79.51 for CUB-200-2011 + EfficientNet).

A per-class analysis was performed for each of the configurations to understand where the gains of the method came from. Table 5.4 shows an example analysis from the scenario of CUB-200-2011 + CSWin-Large. For each of the classes in a cluster, we compare the per-class F1 score of the baseline with the one resulting from the final step of the method. Additionally, we also compare the per-class recall from the baseline and from the specialization step, in order to verify if there is any correlation between the increase in the recall during the specialization step and the overall improvement of the model.

Table 5.4 shows that, after the combination step, the performance for many of the classes in the clusters is improving with respect to the baseline. This shows that the method is indeed improving the performance for the fine-grained classes. This is also confirmed when measuring the evolution

of the top-1 validation accuracy for those classes that do not belong to any cluster, and those that do. Table 5.5 shows how, for this dataset-model pair, the accuracy for the 'generic' classes (those not belonging to any cluster) remains almost intact throughout the method, whereas the accuracy for the 'fine-grained' classes (those that belong to a cluster) is improved by more than two percentage points. This analysis verifies that the method is indeed focusing on improving the classification performance for the classes in the clusters, confirming its suitability for fine-grained problems. It must also be considered that Tables 5.5 and 5.4 show the results from CUB-200-2011 + CSWin-Large, which is the dataset-model pair for which we obtained the greatest performance gains with the original FGFR.

		Step 1		Step 2	Step 3
Cluster ID	**Class**	**Recall**	**F1**	**Recall**	**F1**
1	117	0.80	0.75	0.87	**0.81**
1	129	0.67	0.77	0.70	**0.83**
2	142	0.77	0.79	0.70	**0.81**
2	144	0.83	0.82	0.73	0.81
3	22	0.76	0.80	0.72	0.76
3	23	0.86	0.84	0.82	**0.93**
3	24	0.73	0.72	0.77	**0.76**
4	58	0.17	0.23	0.33	**0.29**
4	63	0.90	0.78	0.90	**0.81**
4	65	0.70	0.60	0.43	0.52
5	36	0.62	0.67	0.62	0.64
5	38	0.55	0.58	0.59	0.57
6	21	0.62	0.74	0.85	**0.76**
6	104	0.95	0.73	0.84	**0.74**
7	66	0.77	0.78	0.77	**0.81**
7	67	0.87	0.83	0.93	**0.84**
8	39	0.83	0.81	0.93	0.84
8	101	0.70	0.74	0.73	0.82

Table 5.4: Detailed per-class performance comparison across the three steps of the pipeline, for the first seven subsets from the case of CUB-200-2011 + CSWin-Large.

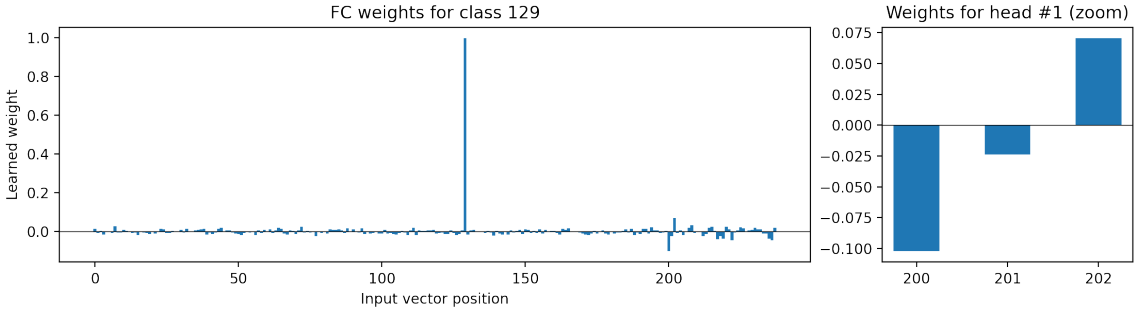
Top-1 validation accuracy	**Baseline**	**FGFR**
'Generic' classes (not in cluster)	94.05	94.07
'Fine-grained' classes (in cluster)	77.63	79.80

Table 5.5: Comparison of the top-1 validation accuracies between the baseline and the final step of the original FGFR, for the two kinds of classes: those that do not belong to any cluster (referred to as 'generic') and those that do ('fine-grained'). Scenario: CUB-200-2011 + CSWin-Large.

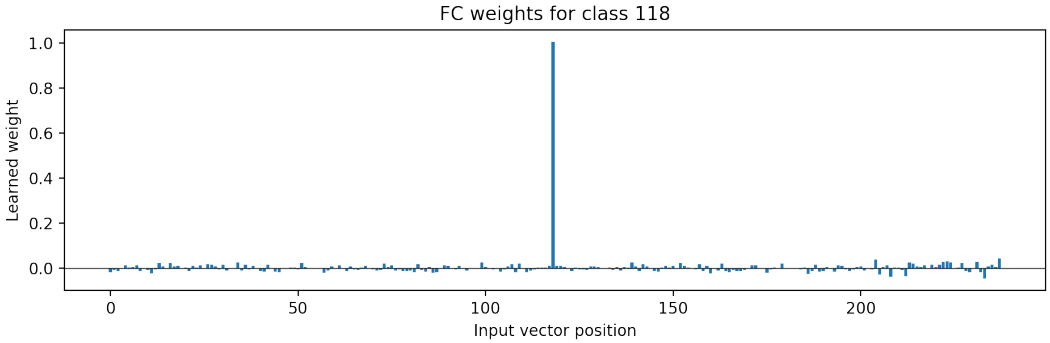
A qualitative analysis of the learned weights for the final FC was performed, in order to estimate the pattern learned by the model to decide which heads to pay attention to. Figure 5.15 shows the values of two different rows from the $C \times S$ weight matrix, where C is the number of classes in the original dataset, and S is C plus the sum of the number of classes in each specialized head. The model learns that, for those classes that belong to a cluster, it should give positive weight to the prediction for that class made by its corresponding head, and give negative weights to the predictions of all other classes within that subset. Given that the weight matrix was initialized with the identity matrix, the weight for the corresponding class in the baseline head is very high, with

a value almost unchanged of 1. Consequently, we believe the model is relying on the baseline to make its predictions, but that it is also using the corresponding head prediction to 'tip the balance' towards the correct class whenever it is faced with a fine-grained class (in those cases where the baseline would potentially have two or more predictions with very similar, low confidence scores).

Even in the case of classes that do not belong to any cluster, the model is learning some patterns for the specialized head predictions. More concretely, we observe it is learning to give positive weights to some of the heads' 'others' predictions. We find this idea intuitive, as it allows the model to be more confident that a certain generic class is indeed not part of any cluster. However, this is not reflected in the quantitative evaluation of the method, as the performance for the generic classes remains almost untouched.



(a) Weight matrix, row 129 (weights to predict class 129). Note how column 129 has a weight of 1 (as it was initialized with the identity matrix). As class 129 belongs to cluster 1 (containing two classes plus 'others'), the corresponding weights of this row to that cluster/head (on the right) reflect the following: negative weight for the 'others' class (left column), negative weight for the other class in the cluster that is not 129 (middle column), and positive weight for the column corresponding to class 129 within the cluster (right column).



(b) Weight matrix, row 118 (weights to predict class 118). Class 118 does not belong to any cluster, so only one column has a strong positive weight, the one corresponding to it from the baseline (column 118).

Figure 5.15: Visualization of some rows of the weight matrix learned for the final FC layer, for the case of CUB-200-2011 + CSWin-Large.

Different initialization methods were tried for the weights of the final FC layer. They were first evaluated by measuring the accuracy and loss obtained by directly performing a forward pass over the validation set, without any training. All of them consisted of different variations of the identity matrix initialization concept. Some ideas included:

- Replacing the zeroes in the identity matrix with -1.

- Initializing the weights corresponding to the specialized heads so that the predictions from these heads are mapped to the actual classes they belong to (an idea analogous to initializing the weights for the baseline head with the identity matrix).

- Assigning negative weights for the 'others' class.

None of these methods (individually or combined with each other) led to an improvement in the results. We found that both the accuracy and the loss were generally worse when we gave more weight to the predictions made by the specialized heads. Training using these initialization methods did not contribute either.

Finally, we noticed that the outputs generated by each of the heads were not being passed to an activation function. Consequently, we tried several activation functions, namely softmax, ReLU, tanh and sigmoid, in order to both increase the expressivity of the model by adding non-linearities, and also to homogenize the range of values generated by each of the heads. None of these experiments led to an improvement in terms of accuracy or loss. We only managed to match the original accuracy of the model (reported in Table 5.3) by using the softmax function. However, this led the loss value to increase in around an order of magnitude, and thus we decided to discard the idea.

5.2.4 Limitations of the Original FGFR

During our the study of the behavior of the original FGFR we identified a series of issues or limitations of the method. We summarize them as follows:

- During the specialization step, the precision of most of the subset classes tends to decrease significantly, given the relatively low recall of the 'others' classes. By contrast, only around half of the subset classes tend to improve their recall during the training of the heads. Although we suspect the issue of the low precision could be partially alleviated during the aggregation step, we believe there is some room for improvement in the training of the specialized heads.

- We found the aggregation step to be notably hard to train. Even with a proper learning rate, the learning of the model is unstable, with accuracy evolution plots that are jagged, with sharp peaks and valleys, with no clear pattern or trend, and characterized by volatile fluctuations which, in some cases, might lead to slight improvements over the baseline performance. We argue that the current aggregation method is failing to leverage the potential of the multi-head model.

- As the authors of FGFR already pointed out [51], we find the method to be computationally expensive to train, both for having to train its three steps sequentially and for significantly increasing the size of the baseline architecture to allow for the multi-head model.

5.3 Results of FGFR+

We present here the results obtained using our improvement of FGFR i.e. the FGFR+ method. Table 5.6 compares the top-1 validation accuracy obtained using the original FGFR and FGFR+, for all the dataset-model pairs that were described in Section 5.1. In order to emphasize the

contribution of the method itself, we report our results without using TTA. For each configuration, we also report the results obtained with the corresponding baseline model.

Dataset	Model	Baseline	FGFR	FGFR+
CUB-200-2011	EfficientNet-B0	79.44	79.51	**81.81**
	CSWin-Large	90.21	90.40	**90.87**
Stanford Cars	EfficientNet-B0	91.78	91.89	**92.87**
	CSWin-Large	93.76	93.84	**94.39**
Food-101	EfficientNet-B0	87.56	87.68	**88.09**
	CSWin-Large	94.10	94.10	**94.16**

Table 5.6: Top-1 validation accuracy obtained using FGFR+ (our proposal), for all dataset-model pairs considered. For each pair, we also report the accuracy obtained with the baseline model and the original FGFR.

Our proposed method consistently improves upon the results obtained using the original FGFR. Unlike the original method, FGFR+ proves to be capable of improving with respect to the baseline in all scenarios. Whereas the original method would derive in improvements of around 1 or 2 decimal points in accuracy over the baseline, our proposed method tends to improve, on average, more than half a point. We even observe that for some configurations the improvement goes beyond one decimal point (+2.3 percentage points for CUB-200-2011 + EfficientNet-B0, and around +1 for Stanford Cars + EfficientNet-B0). It must be considered that the training of the aggregation step of our approach is noticeably more expensive than the original one, as it involves training the complete model rather than just the final FC layer. Nevertheless, we believe that our method's improvements in terms of classification performance are sufficiently large so as to compensate for the longer training time.

We notice significant improvements for all datasets and models, which suggests that FGFR+ can generalize well to different baseline model architectures and domains. However, we observe the biggest improvements when using the EfficientNet-B0 model, which could hint that the method might be more effective for lower capacity models. Regarding the datasets, our approach leads to significant improvements in all of them, although we observe that the gains are smaller for the Food-101 dataset. This was somewhat expected, as food is a particularly challenging domain within the field of fine-grained image analysis.

5.3.1 Comparison with Other Methods

We also compare the results of our approach with those reported by the current state-of-the-art methods for each of the three datasets. For our method, given that the CSWin-Large backbone consistently outperforms the EfficientNet-B0 backbone, we only include the results obtained with the former. These results are reported in Tables 5.7a, 5.7b and 5.7c.

We find our method to be unable to match the results reported by the best perfoming methods for each dataset. In the case of CUB-200-2011, our model's top-1 validation accuracy is just under 2 percentage points behind the top method, PIM [9], and would not make it either into the top 5 in terms of accuracy. Similarly, for Stanford Cars our method falls 1.3 points behind CAP [3], the top method in terms of top-1 accuracy. However, in the case of the Food-101 dataset, our

Method	Top-1
PIM [9]	**92.8**
DCAL [69]	92.0
SR-GNN [4]	91.9
ViT-SAC [12]	91.8
SIM-Trans [54]	91.8
Ours (CSWin)	90.9*

(a) CUB-200-2011.

Method	Top-1
CAP [3]	**95.7**
AttNet & AffNet [22]	95.6
CAL [50]	95.5
Inceptionv4 [49]	95.4
API-Net [70]	95.3
Ours (CSWin)	94.4*

(b) Stanford Cars.

Method	Top-1
CAP [3]	**98.6**
Efficient Adaptive Ensembling [7]	96.9
Grafit [57]	93.7
GPipe [28]	93.0
EfficientNet-B7 [55]	93.0
Ours (CSWin)	94.2*

(c) Food-101.

Table 5.7: Accuracy comparison with the top five state-of-the-art approaches for each dataset, filtering out those that make use of extra training data. The results for our method are marked with an asterisk (*) as we report our results as validation results, so they are not directly comparable with those reported by other methods.

method would rank third (to our best knowledge), although the difference with the first method (once again, CAP) is quite significant, of around 4%.

It must also be taken into account that our method can theoretically be applied on top of any backbone architecture, so our results could potentially be improved if we used a more performant baseline model.

5.4 Analysis of FGFR+

In this section, we explore the effect of a variety of modifications we implemented on top of the original FGFR, as part of the development of FGFR+. Each of the following sections covers different ideas we experimented with: the effect of combining head embeddings for the aggregation step (Section 5.4.1), the use of self-attention as a head-output combination method (Section 5.4.2), using multi-task weighting losses for the specialization step (Section 5.4.3) and the effect using focal loss (Section 5.4.4).

5.4.1 Effect of Combining Head Embeddings for the Aggregation Step

In this part of the analysis, we evaluate the effect of bypassing the classifier of each of the heads for the aggregation step, so that the final FC layer directly takes as inputs the embeddings of the heads. We explore multiple ways of combining the embeddings generated by the different heads. We also measure the impact of unfreezing different parts of the model during the training of the aggregation step. In order to be able to experiment in an agile way, we made use of the EfficientNet-B0 model as the baseline architecture (given that it is significantly smaller than CSWin-L), and we tested it on the smallest datasets, namely CUB-200-2011.

We compare all of our results against both the baseline model and those obtained using the original FGFR. Table 5.8 features the relevant metrics for each of these two models.

Model	**Overall Top-1**	**Generic Top-1**	**Cluster Top-1**
Baseline	79.44	83.63	65.59
Original FGFR	79.51	83.73	66.19

Table 5.8: Overall top-1 (for all classes in the dataset), generic top-1 (accuracy for classes that do not belong to any cluster) and cluster top-1 (accuracy for classes that do belong to a cluster) validation accuracy for the baseline and complete original FGFR, after the aggregation step. Scenario: CUB-200-2011 + EfficientNet-B0.

We performed experiments to evaluate the effect of two different kinds of modifications:

- The method used to combine the embeddings generated by each of the heads. We experimented with two methods: *concatenating* the embeddings, and calculating their *mean*. Assuming that there are H heads and that each head generates an image embedding of length E, concatening all the embeddings would result in feeding the final FC connected layer with a vector of length $H \cdot E$ per image. On the other hand, averaging the embeddings would result in a final vector of length E that would be passed to the final FC layer.

- The fraction of the model that is trained during the aggregation step. During this step, the original FGFR only trained the final FC layer. We explore two additional approaches: unfreezing the whole model (so that the backbone and the heads are also trained during this step) and unfreezing only the heads, keeping the backbone frozen.

We experimented with all different combinations of the aforementioned embedding combination and model unfreezing approaches, and we report the obtained results in Table 5.9. It must be noted that, for each combination-unfreezing approach pair, we performed a grid search over the learning rate to find the best model for each configuration.

Combination	**Unfreeze**	**Overall Top-1**	**Generic Top-1**	**Cluster Top-1**
Concatenation	Backbone + heads	80.45	84.73	67.62
	Heads	79.70	83.96	68.02
	Nothing	79.31	83.62	67.97
Mean	Backbone + heads	**81.53**	**85.45**	**69.74**
	Heads	80.43	84.59	68.89
	Nothing	78.56	82.63	68.15

Table 5.9: Validation results obtained using different embedding combination methods and unfreezing different parts of the model (results for CUB-200-2011 + EfficientNet-B0).

We can extract multiple conclusions from Table 5.9 if we compare its results to those reported in Table 5.8 (baseline and original FGFR). First, the embedding combination approach is not capable of improving with respect to the baseline if both the backbone and the heads remain frozen during this training step. In this sense, the original FGFR seems to be performing better, as it does not require any of these parts to be unfrozen in order to outperform the baseline results.

However, by just unfreezing the heads, we already obtain results significantly higher than those

reported for the original FGFR, for both combination methods (79.70 and 80.43 against 79.51). If we let the whole model train during this step (jointly training the final FC layer and fine-tuning the backbone and the head), the results improve even more. We hypothesize that, in order for the combination of the embeddings to be effective, we need to give the network some margin to modify the way the embeddings are generated (by unfreezing it), so that they are optimized for the combination method used. We believe this is particularly important in the case of the *mean* combination method.

A significant difference can be observed between the *concatenation* and *mean* combination methods, where the latter leads to substantially higher results whenever some parts of the model are unfrozen. By using the mean combination method and unfreezing the whole model during the aggregation step, we achieve a top-1 accuracy of 81.53%, an improvement of more than 2% with respect to the baseline (compared to the 0.07% of improvement by the original FGFR). By analyzing the performance for the generic and the cluster classes, it can be seen that this configuration improves both metrics significantly when compared to the baseline: an improvement of around 2% for the generic classes, and more than 4% for the fine-grained classes. This shows that this method is not only capable of improving for the fine-grained classes, but also for the generic ones. Moreover, the greatest relative improvement corresponds to the cluster classes, proving that this new approach is still suitable for fine-grained problems.

By analyzing the evolution of the evaluation metrics, we observe a significant difference with respect to the original model (see Figure 5.16). Not only the use of embeddings facilitates the training of this step, but it also makes the learning more stable and it achieves a consistent performance that is superior to that of the baseline. In the original method, these plots were much more jagged, and the improvements over the baseline could typically be attributed to casual peaks, showing no consistent learning. With the new aggregation method, the upward trend is much more clear, showing that the model is indeed performing some meaningful learning.

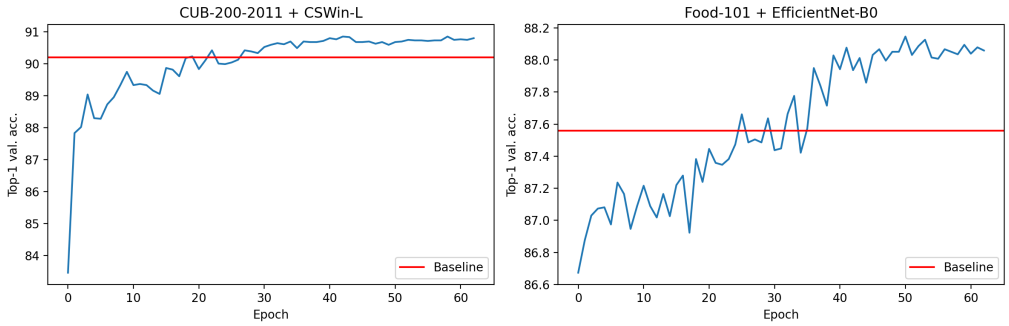


Figure 5.16: Top-1 validation accuracy evolution charts for two configurations, for the training of the aggregation step using head embeddings, combining them with the mean, and keeping the whole model unfrozen.

To make the comparison as fair as possible, we also performed experiments with the original aggregation method, following the new model unfreezing policies, as one could think that the new approach only obtains better results because of the larger portion of the model that is being trained. Table 5.10 shows that, even if the results improve slightly by unfreezing the whole model, they are still very distant from those reported with the new aggregation method, showing the benefits of directly dealing with head embeddings rather than with the predictions made by them. Unlike

our new approach, the original one barely benefits from unfreezing previous parts of the model.

Unfreeze	Overall Top-1	Generic Top-1	Cluster Top-1
Backbone + heads	79.55	83.70	66.74
Nothing	79.51	83.73	66.19

Table 5.10: Results obtained after training the original aggregation step, unfreezing different parts of the model (CUB-200-2011 + EfficientNet-B0).

Table 5.11 shows the results obtained using this new aggregation method (unfreezing the whole model and combining the head embeddings using the mean) for all dataset-model pairs. The better performance of this new approach is confirmed as it improves the overall top-1 validation accuracy in all cases.

Dataset	Model	Baseline	Original aggregation	New aggregation
CUB-200-2011	EfficientNet-B0	79.44	79.51	**81.53**
	CSWin-Large	90.21	90.40	**90.82**
Stanford Cars	EfficientNet-B0	91.78	91.89	**92.85**
	CSWin-Large	93.76	93.84	**94.28**
Food-101	EfficientNet-B0	87.56	87.68	**88.15**
	CSWin-Large	94.10	94.10	**94.14**

Table 5.11: Comparison of the top-1 validation accuracy obtained with new aggregation method (combine embeddings with mean, unfreeze everything) and the original one, for all dataset-model pairs.

5.4.1.1 Evaluating the Contribution from the Baseline

Given that the new best aggregation method involves unfreezing the whole model, we would like to know how much of the improvement of this new approach can be attributed to the the mere improvement of the baseline. To that end, we take the new weights for the backbone and the baseline head resulting from the training of the new aggregation step, we freeze them, and train a linear classifier on top of them. With this, we evaluate how much the baseline has improved with respect to the initial step of the method. We report in Table 5.12 the results obtained with the original baseline, the new modified baseline, and the complete method after training the aggregation step.

Model	Overall Top-1	Generic Top-1	Cluster Top-1
Complete method	81.53	85.45	69.74
New baseline	80.26	84.42	67.18
Original baseline	79.44	83.63	65.59

Table 5.12: Validation metrics for the original baseline, the new modified baseline and the complete method using the new aggregation method (results for CUB-200-2011 + EfficientNet-B0).

We observe that, after the aggregation step, the baseline has improved significantly, around 0.75% with respect to its initial version. The initial baseline was trained carefully and performing an

extensive grid search over hyperparameters, and we were not capable of achieving a top-1 accuracy higher than 79.44 using EfficientNet-B0 for CUB-200-2011. With this method, the baseline is improving significantly, suggesting that the new aggregation method is also facilitating the further improvement of the baseline. Nevertheless, the performance of the new baseline is still low compared to the results obtained with the complete model, which proves the importance of the specialized heads.

Additionally, we perform an experiment regarding the dependency of the specialized heads on the baseline head. We train the new aggregation step, but we ignore the embeddings generated by the baseline head, only combining those from the specialized heads. With this, we aim to discover 1) how much generic information the specialized heads are capable of learning, and 2) how much the final performance depends on the information provided by the baseline. We find that the final performance of the model decreases significantly, falling around one percentage point below the top-1 accuracy of the baseline. This shows that, even though the specialized heads are obviously learning some information regarding the generic classes (otherwise the performance would be much lower), this generic information is still not as rich as the one provided by the baseline head, hence its importance for the good performance of the complete model.

5.4.2 Effect of Self-Attention in the Aggregation Step

Building on top of the findings reported in Section 5.4.1, we wonder whether there could be a better method for combining the head embeddings. In particular, we hypothesize that, ideally, the final FC layer should only take into account the information provided by the baseline head and the head corresponding to the cluster the current image belongs to (in case it belongs to any subset). We argue that all other information could be misleading the final prediction, and that a head *filtering* method could help improve the final performance of the model. We hypothesized that the task of deciding which heads to focus on could be too complex for a simple FC layer.

Based on the recent success of attention-based models, we evaluate the use of a Multi-Head Attention module [59] as a method to combine the embeddings generated by the heads. In the ideal case, we would like the attention module to learn which heads to pay attention to at each time, depending on the subset the image at hand belongs to. The resulting output embeddings from this module would then be enriched with information coming only from the relevant heads, simplifying the task of the final FC layer of distinguishing the class that the current embedding belongs to.

Figure 5.17 illustrates the proposed architecture. For each image, all the embeddings generated by the heads (both the specialized and the baseline head) are passed to the multi-head attention module. This module takes as input a sequence of embeddings and outputs an embedding sequence of the same length. Each of the output embeddings can be understood as a weighted sum of the original input embeddings, depending on the attention scores that the module assigns to each of the input embeddings to generate the current one.

In order to guarantee the proper training of the network, and also to ensure that no information is lost during the attention-based combination process, we add a residual connection around the attention module. In this way, each output embedding contains both information from its original, corresponding embedding, as well as enriched information from other heads as a result of the attention process. We use a multi-head attention module with eight internal parallel attention

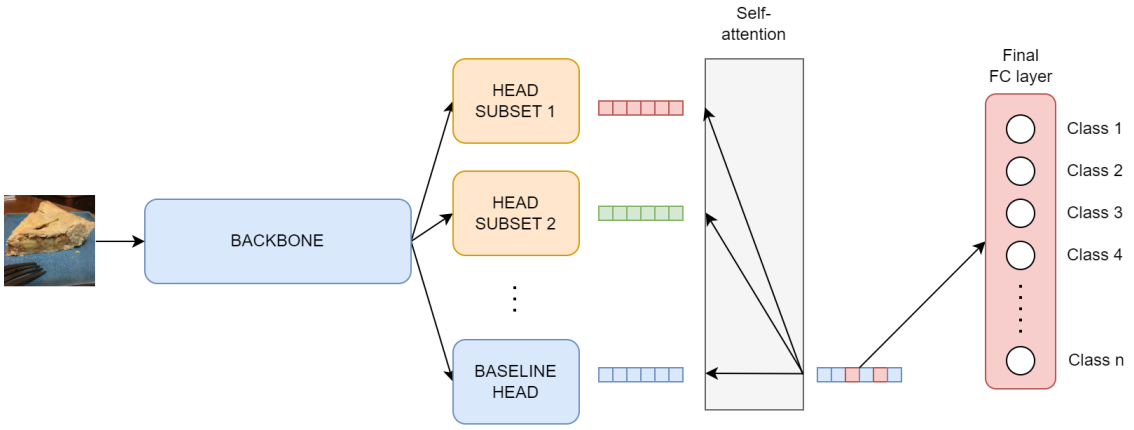


Figure 5.17: Diagram illustrating the integration of multi-head attention into the aggregation step. In this case, after the attention we only take the embedding corresponding to the baseline head and pass it to the final FC layer.

heads, following [59].

Our first approach simply takes the output embedding corresponding to the baseline head (as shown in Figure 5.17) and passes it to the final FC layer, ignoring the embeddings of all other heads. The rationale behind this process is that the baseline embedding already contains a decent generic representation for all classes, and that it is only missing some fine-grained details about the current cluster to be able to properly distinguish the current class. Consequently, we argue that we could use the attention module to 'enrich' the information of the baseline embedding with relevant information from the corresponding specialized head. Ideally, the resulting embedding would contain both generic information necessary to distinguish the current class from the generic ones (coming from the baseline head), and also fine-grained information essential to distinguish between the classes in the current cluster (coming from the corresponding specialized head).

With this setting, we train the aggregation step of the method, while also evaluating the effect of unfreezing different parts of the model. The obtained results are shown in Table 5.13. Once again, we mainly report the results obtained for the dataset CUB-200-2011 using the EfficientNet-B0 model as the baseline, as it allowed us to experiment in a more agile way.

Unfreeze	**Overall Top-1**	**Generic Top-1**	**Cluster Top-1**
Backbone + heads	**80.57**	**84.42**	68.14
Heads	79.70	83.80	**68.44**
Nothing	79.36	83.39	67.69

Table 5.13: Validation results obtained using the multi-head attention module as the head embedding combination method, only keeping the output embedding corresponding to the baseline head. Results shown for the configuration CUB-200-2011 + EfficientNet-B0.

We make multipe observations from the results shown in Table 5.13. On the one hand, we see that, without unfreezing any previous parts of the model, this approach for the aggregation step is not capable of matching the performance of the baseline (see Table 5.8 for the baseline results). However, when either the heads or the whole model are unfrozen, the model starts to outperform

both the baseline and the original aggregation method proposed in [51]. Nevertheless, this performance is still far from the one reported in Table 5.9, falling around one accuracy percentage point behind the best configuration we found for this dataset-model pair. We observe the same pattern we saw while evaluating the different embedding combination methods in Section 5.4.1: the accuracy for both the generic and the fine-grained classes is improving, but the improvement for the fine-grained classes is significantly larger.

To better comprehend the learned behavior of the multi-head attention module, we visualize the attention scores that the module assigns to each of the heads. Given that the attention scores will be different depending on the image, we store the scores generated for all images in the validation set, and we group them by the subset they belong to (we also group together those images that do not belong to any cluster). We plot the average attention scores for each of the subsets in Figure 5.18.

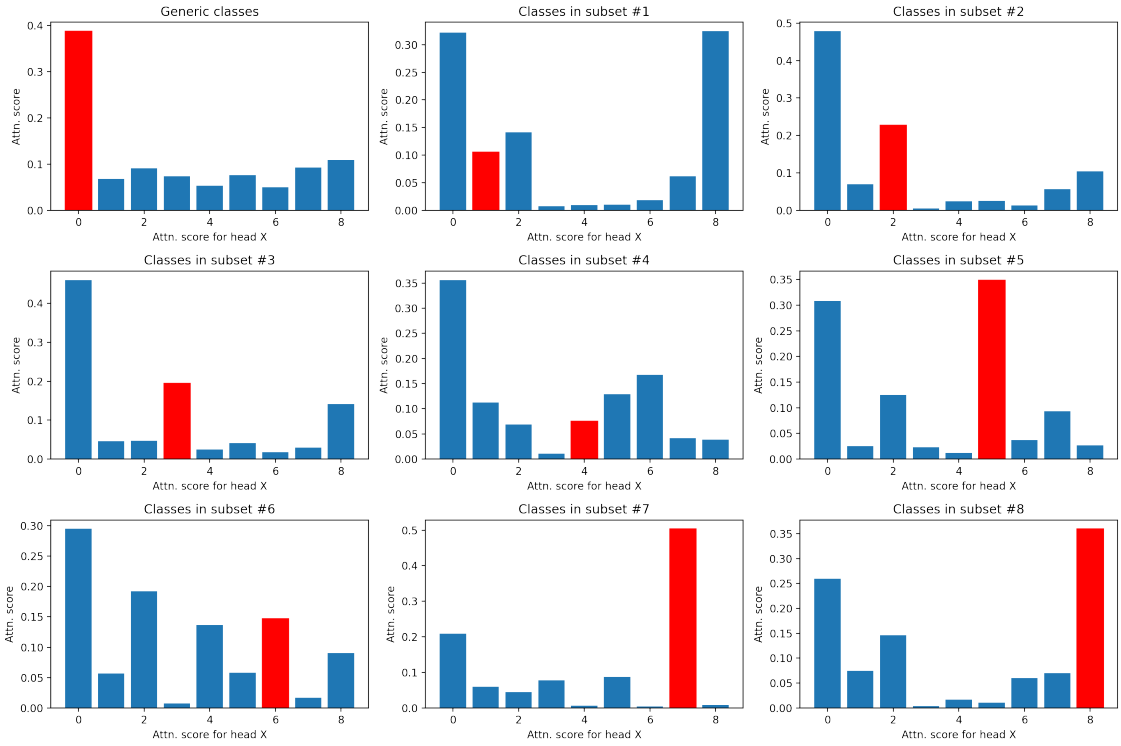


Figure 5.18: Attention scores generated by the best configuration (unfreezing the whole model), for CUB-200-2011 + EfficientNet-B0. Head 0 corresponds to the baseline head. The bar in red represents the head correponding to the cluster. Example: the plot on the top right corner shows the average attention scores for all images belonging to the second cluster. The average attention scores in the plot reflect that the model mainly focuses on the baseline head and the second head (the 'correct' one, in red) to generate the resulting embedding.

The visualizations from Figure 5.18 suggest that the attention module is indeed learning the behavior we had originally thought of: in most of the cases, it is capable of identifying which is the head corresponding to the cluster of the current image, and consequently assigns a higher attention score to it. We also observe that the module tends to assign a high attention score to the baseline head, which we expected, as it can contain useful information for any class.

However, we notice that, in some cases, the module is giving a very high attention score to a head

that is not the one corresponding to the current cluster (the most clear example is the plot for the subset 1 in Figure 5.18). By analyzing the images belonging to the two clusters involved (the actual one and the one that got the highest attention score, subset 8), we observe that they contain very similar objects (in this case, 'black birds'), which could also be potentially indistinguishable by a non-expert human. Consequently, we believe that the attention module is either confusing these two clusters with one another, or it has learned that the embedding from the 'wrong' head is more useful to identify the current class than the one from the head it would correspond to.

In any case, the quantitative results obtained using this method are considerably worse than those obtained using other combination methods, namely the mean. We argue that there might be two main reasons for this. On the one hand, with the current approach, it might be a complex task to compress all the relevant information from the different heads in to a single embedding (note that the input and output embeddings of the attention module have the same dimensionality). On the other hand, we also consider the possibility of the attention module giving importance to heads that are indeed irrelevant (as we previously commented with the example of the 'black bird' subsets), potentially adding some kind of noise to the final output embedding.

5.4.2.1 Combining Attention Embeddings

In order to tackle the first of the problems mentioned, we experiment with an alternative method that does not discard the output embeddings corresponding to the specialized heads, but which takes all of them into account and combines them using the embedding combination approaches discussed in Section 5.4.1. Figure 5.19 describes the new approach. Using this method and thanks to the residual connection, the information from the original embeddings is guaranteed to make it past the attention module, plus the extra benefits of self-attention, which we already showed that is capable of identifying the most relevant heads for each image.

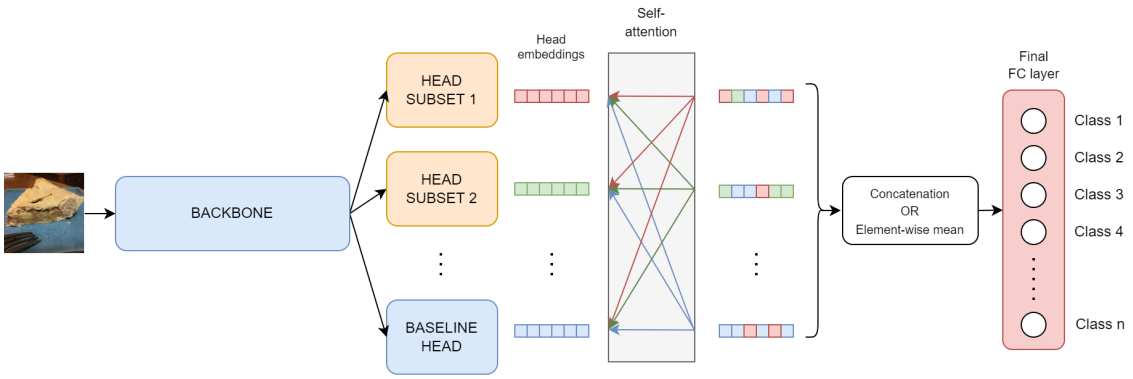


Figure 5.19: Diagram illustrating the method combining the output embeddings from the attention module, using either the mean or the concatenation method.

We experimented using both the element-wise mean and the concatenation as the embedding combination methods. This approach is exactly the same as the one described in Section 5.4.1, but modifying the embeddings using attention, prior to their combination. Again, we experiment with different ways of unfreezing the model too. The results obtained using this model are shown in Table 5.14.

In this case, we see that the three methods (concatenation, mean, and just taking the embedding

Combination method	Unfreeze	Overall Top-1	Generic Top-1	Cluster Top-1
Concatenation	Backbone + heads	**80.77**	**84.92**	**68.62**
	Heads	80.00	83.95	67.27
	Nothing	79.67	83.77	67.62
Mean	Backbone + heads	80.60	84.79	66.47
	Heads	79.74	83.90	67.21
	Nothing	79.17	83.50	66.26
Baseline embedding	Backbone + heads	80.57	84.42	68.14
	Heads	79.70	83.80	68.44
	Nothing	79.36	83.39	67.69

Table 5.14: Results obtained using self-attention followed by the combination of the embeddings using either the concatenation or the mean. We also include the results obtained using our initial approach with self-attention ('Baseline embedding', originally reported in Table 5.13).

corresponding to the baseline head that the attention generates) achieve similar performances when trained while unfreezing the whole model. We also observe that, for the first time since we started experimenting with embedding combination methods for the aggregation step, we managed to outperform the baseline without unfreezing any previous parts of the model (with concatenation).

Given the similar results obtained for the three methods unfreezing the complete model, we made use of statistical significance tests to determine whether there was any real difference between their performance. We relied on the Almost Stochastic Order (ASO) test [2]. Using ASO with a confidence level $\alpha = 0.05$, we found the top-1 validation accuracy distribution of the mean method based on five random seeds to be stochastically dominant over both the concatenation and the original (baseline embedding) approach ($\epsilon_{min} = 0.05$ and 0.01, respectively).

Nevertheless, these results are still far behind those obtained with the more simple approach of just computing the mean of the embeddings, without using any kind of attention mechanism. However, we believed that this line of research was promising, especially given the favorable findings from Figure 5.18.

We experimented with different aspects of the approach, such as using positional encodings, or testing different values for the number of attention heads within the multi-head attention module. None of them led to an improvement of the results reported in Table 5.14.

5.4.2.2 Analyzing the Attention Scores of the New Combination Approaches

To try to understand the behavior of the attention module when combining its outputs using either the mean or the concatenation operations, we opted to visualize the generated attention scores. Unlike the case of Figure 5.18, where there would only be one output embedding from the attention module, with the new approaches the attention generates as many embeddings as there are heads, and the attention scores distribution could potentially be different from one embedding to another.

We start by visualizing the attention scores used to generate the output embedding corresponding to the baseline head, averaging the attention scores for images that belong to the same subset (as in Figure 5.18). The attention scores are shown in Figure 5.20, for the case of the concatenation

method, unfreezing the whole model.

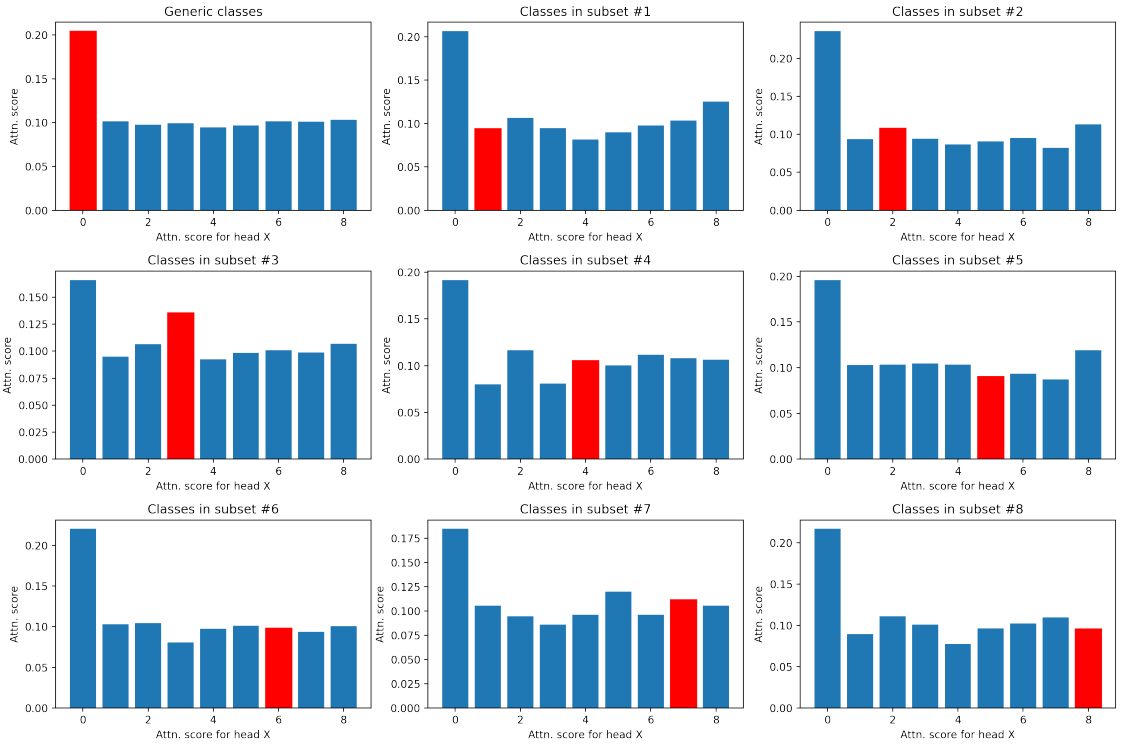


Figure 5.20: Visualization of the attention scores using the concatenation method and unfreezing the whole model during the aggregation step (CUB-200-2011 + EfficientNet-B0). The scores visualized are those used to generate the output embedding corresponding to the baseline head.

It can be seen that the attention patterns in Figure 5.20 are very different from those shown in Figure 5.18. In this case, the attention module is failing to correctly highlight the corresponding head in each case. In most of the cases, the baseline is receiving most of the attention (which is expected), but all of the remaining attention is almost equally split among the specialized heads. This behavior remains consistent when evaluating the attention scores used to generate other output embeddings.

By assigning all the specialized heads the same amount of attention, we believe that the contribution of the attention mechanism is essentially reduced to noise, as it is failing to correctly identify the most relevant aspects of each of the heads to determine whether they could be useful or not. Moreover, we believe that the relatively good performance of the model can be almost entirely attributed to the information coming from the residual connection, rather than the information generated by the attention module itself. This becomes even more plausible when considering that the attention patterns used to generate the different output embeddings are practically always the same. That is, we believe that the resulting embeddings are just the original embeddings (the input of the attention module) plus a constant vector that, as we mentioned, could potentially be considered noise. This idea is illustrated in Figure 5.21. This could explain the inferior performance of these approaches when compared to those described in Section 5.4.1.

In order to test our hypothesis, we experiment with removing the residual connection, forcing the attention module to learn meaningful patterns to effectively combine the embeddings. We observe

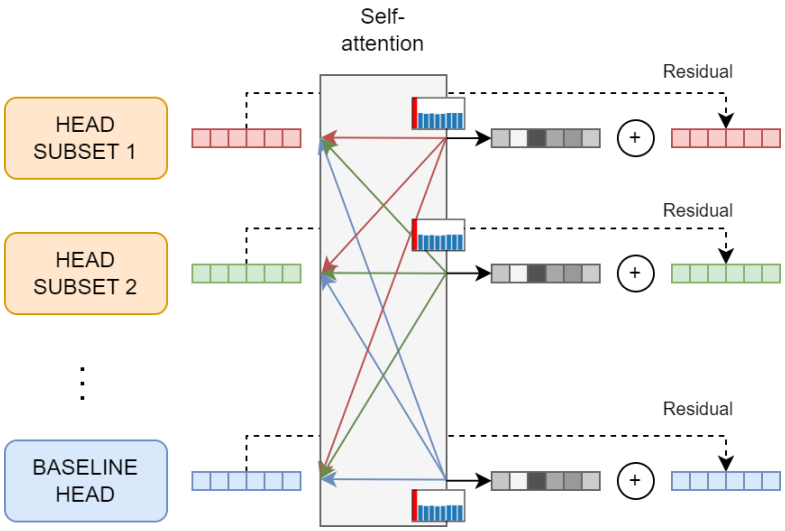


Figure 5.21: Illustrative diagram explaining how, by keeping the attention scores distribution constant among the different output embeddings, the direct output of the attention module is constant throughout all the output embeddings. Moreover, knowing that the attention distribution tends to assign all specialized heads the same amount of attention, we believe this constant vector does not contain much useful information for the final prediction. We believe that the most useful information comes from the residual connection, rather than the attention itself.

a significant deterioration in the top-1 accuracy of the model of around two percentage points below the accuracy of the baseline model. By analyzing the corresponding attention scores, it can be seen that the attention patterns do not change much from those showed in Figure 5.20, generally assigning the same scores to all heads except for the baseline head, which typically receives more attention. This confirmed that the performance of the model largely depends on the residual connection.

5.4.2.3 Experimenting with Transformers

As a final attempt to leverage the benefits of self-attention in the multi-subsets method, we explore replacing the multi-head attention module with full transformer encoders [59]. Our rationale behind these experiments is to verify whether increasing the complexity of the head combination mechanism could help overcome the problems presented in the previous sections.

Given that the inputs and outputs of the multi-head attention module and the transformer encoder are analogous, the modification of the architecture becomes trivial. We focus on exploring the effect of adding different numbers of transformer encoders. For these experiments, the outputs generated by the encoder blocks are combined by computing their mean, as this was the combination method that proved to be superior using the multi-head attention module. We report the obtained results in Table 5.15.

We find that increasing the number of encoders generally results in a decrease in performance. By stacking transformer encoders we are also increasing the number of attention modules involved, which we hypothesize makes the amount of noise in the final output embeddings more significant, given that with each attention module some noise is added.

Eventually, after an extensive experimentation process, we finally discarded the use of any kind

Encoder layers	Unfreeze	Overall Top-1	Generic Top-1	Cluster Top-1
1	Backbone + heads	**80.76**	**85.05**	67.60
	Heads	80.01	84.29	67.75
	Nothing	79.70	84.11	67.55
2	Backbone + heads	80.43	84.63	68.01
	Heads	79.65	83.80	**68.36**
	Nothing	79.48	83.68	68.23
4	Backbone + heads	80.29	84.43	67.14
	Heads	79.58	83.69	67.62
	Nothing	79.50	83.75	66.73

Table 5.15: Results obtained with different numbers of transformer encoders, combined with the mean method, unfreezing different parts of the model. The results correspond to the CUB-200-2011 dataset with an EfficientNet-B0 baseline architecture.

of attention mechanisms for the aggregation step, as its performance was consistently inferior to the much simpler approach of just taking the head embeddings and computing their mean (what ended up becoming our proposed new aggregation step).

5.4.3 Effect of Multi-Task Loss Functions for the Specialization Step

We also explored ways of improving the specialization step. We mostly focused on the optimization process of the multi-task training, which, in the original FGFR, is performed by simply computing the average loss from the specialized heads. We tested some existing multi-task learning optimization approaches to evaluate whether they could enhance the learning of the heads. In particular, we experimented with 1) using uncertainty to weigh losses [32] (we refer to it as Uncertainty Weighted Loss, UWL), and 2) the method introduced in [36], which is just a slight modification on top of UWL, which we refer to as Automatic Weighted Loss (AWL).

Given that the two optimization approaches share their overall structure (both are methods that formulate a single aggregated loss function as a weighted sum of the individual losses, where the weights are learnable parameters of the model), we trained both using the same configuration. The parameters of these models were trained using the same type of optimizer but with a constant learning rate value, unlike the rest of the parameters of the model.

We first report the results obtained for the specialization step, analyzing whether the new approaches improve the learning of the specialized heads. We only report the results for AWL, as we found the two methods to yield almost identical results in all cases, possibly because their formulations are very similar (AWL is presented as an improvement of UWL that prevents the loss value from becoming negative, an issue we did not observe during our experiments). Table 5.16 shows the average improvement ratio of the per-class precision, recall and F1 score for all the classes that belong to a cluster, between the original specialization step and the new one trained with AWL, for two dataset-model pairs.

Table 5.16 reveals promising results, as all the average per-class metrics seem to be improving when using AWL, especially the precision, whose improvement is substantial (36% and 11%). In the table, a ratio greater than one implies that the given metric is higher for the new, AWL-

Scenario	Precision imp. ratio	Recall imp. ratio	F1 imp. ratio
CUB-200-2011 + EfficientNet-B0	1.36	1.03	1.28
Stanford Cars + EfficientNet-B0	1.11	1.01	1.07

Table 5.16: Average improvement ratio for the validation per-class precision, recall and F1 scores of all classes that belong to any cluster, when using AWL.

optimized approach. A ratio smaller than one means that the given metric was higher when trained without AWL, by just computing the average head loss. The reported results correspond to those obtained using an EfficientNet-B0 baseline architecture for the CUB-200-2011 and Stanford Cars datasets. An in-depth analysis of the confusion matrices generated by the heads trained with AWL showed that the initial problem stated in Section 5.2.2, where the per-class precision of the cluster classes would decrease given the low recall for the 'others' class, is partially mitigated using these approaches. We observe that with either AWL or UWL the heads generally improve their classification performance for the 'others' class, increasing the number of true positives.

Based on these promising results, we train the new aggregation step (taking the head embeddings and combining them using their mean, as in Section 5.4.1) on top of the learned heads using AWL, to verify whether the clear improvement in the specialization step derives in an improvement also in the aggregation step results. We report an example of the obtained results for a particular configuration in Table 5.17. The reported results are obtained using the new aggregation method, and correspond to the Stanford Cars + EfficientNet-B0 scenario.

Method	Overall Top-1	Generic Top-1	Cluster Top-1
Aggregation w/ mean, step 2 w/ average loss	92.85	**94.79**	75.25
Aggregation w/ mean, step 2 w/ AWL	**92.89**	94.77	**76.42**

Table 5.17: Aggregation step validation performance when training the specialization step using the average of the head losses, and when using AWL.

We observe an improvement for the final overall top-1 accuracy when the specialization step (step 2) is trained to optimize the AWL loss. We notice a significant improvement on the classification of the cluster-classes, even though the top-1 accuracy for the generic, non-cluster classes remains largely the same. This behavior was expected, given that we believe that AWL will mostly affect the performance for the cluster classes, rather than the generic ones (by keeping the baseline head frozen during the specialization step).

We further evaluate the AWL-based training of the heads by applying it to all dataset-model pairs considered within our project. For each dataset-model pair, we take the existing trained baseline and we train the specialization step using AWL, to then train the new aggregation step. The obtained results are reported in Table 5.18.

We observe that the use of AWL for the specialization step leads to an overall improvement of the complete method for five out of the six considered scenarios. However, in many cases, the

Dataset	Model	Top-1 w/ new aggregation	Top-1 w/ AWL + new aggregation
CUB-200-2011	EfficientNet-B0	81.53	**81.81**
	CSWin-Large	90.82	**90.87**
Stanford Cars	EfficientNet-B0	92.85	**92.87**
	CSWin-Large	94.28	**94.39**
Food-101	EfficientNet-B0	**88.15**	88.09
	CSWin-Large	94.14	**94.16**

Table 5.18: Top-1 validation accuracy obtained for all dataset-model configurations, using the original specialization step (left column) and the one optimized with AWL (right column). Each of the reported metrics corresponds to the overall top-1 accuracy obtained after training the three steps of the method.

improvement amounts to less than a decimal point of accuracy. Even if it is small, the improvement of including AWL into the model seems to generalize to all models and datasets, and given that its inclusion in the method pipeline does not imply any additional computational burden, we opted to include it in our final method proposal.

5.4.4 Exploring Other Loss Functions

We experimented with replacing cross-entropy with a different loss function that could lead to a general improvement of the model. Due to temporal and computational restrictions, we were only able to experiment with one other loss function, focal loss [37]. We started experimenting with this loss function to perform the initial step of the method, the training of the baseline, with the goal of verifying whether the use of this new loss function could lead to significant gains in performance. We performed experiments for all three datasets using the EfficientNet-B0 model, testing different values for the γ hyperparameter, and fine-tuning the learning rate for each configuration. We report the obtained results in Table 5.19.

The results show that the use of focal loss with $\gamma > 0$ leads to a decrease in performance for two of the three datasets. The only datasets which benefits from the use of focal loss is Stanford Cars, which could be expected, as focal loss was specifically designed to tackle imbalanced classification tasks, and we already described in Section 5.1.1 that Stanford Cars is significantly more imbalanced than the other two datasets. Even in this case, the absolute improvement in terms of accuracy amounts to less than half a decimal point.

With these results at hand, we opted not to continue experimenting with focal loss for the rest of the steps of the method, given that we were running short on time and resources, and because the obtained results did not suggest that major improvements would be obtained by continuing experimenting with this loss function. Nevertheless, we believe that the specialization step of the model could be the one benefiting the most from being trained to optimize a loss function such as focal loss, as it inherently presents strong data imbalance (which focal loss is designed to tackle). Other promising single-task loss functions for the specialization step could be those based on contrastive learning, forcing each of the heads to learn representations that correctly discriminate between the classes within their cluster.

Dataset	γ	Overall Top-1
CUB-200-2011	0.0	**79.44**
	0.5	79.10
	1.0	78.58
	2.0	78.80
	4.0	78.55
Stanford Cars	0.0	91.78
	0.5	91.54
	1.0	91.58
	2.0	**91.82**
	4.0	91.57
Food-101	0.0	**87.56**
	0.5	84.88
	1.0	87.53
	2.0	87.13
	4.0	86.89

Table 5.19: Validation results obtained after training an EfficientNet-B0 baseline for all datasets, with different values for γ. Note that the case of $\gamma = 0$ corresponds to the use of standard cross-entropy.

5.4.5 Limitations of FGFR+

Although we showed that FGFR+ clearly outperforms the original FGFR, we can still identify several aspects of it that are limiting its potential and its wide-spread applicability:

- Even though the new specialization step mitigates the problem of the subset classes having low precision, we still believe there is room for improvement regarding the learning of the heads, which are the cornerstone of the method.
- Unlike the original one, our proposed aggregation method is only capable of improving upon the baseline when previous parts of the model (backbone or heads) are unfrozen, increasing the temporal and computational requirements needed to train the complete method.
- FGFR+ cannot match the performance of other state-of-the-art approaches. Although the performance of our method could be further boosted by changing the backbone architecture, the same can be said for other state-of-the-art approaches, such as CAP [3] or PIM [9].
- Our proposed modifications do not alleviate the high computational cost of training the complete, multi-step method.

Chapter 6

Conclusions and Future Work

In this work, we focused on replicating, analyzing and improving the original FGFR introduced in [51]. This method was mainly designed for food recognition, so our first goal was to explore how generic and valid the method is to address the problem of fine-grained recognition in other types of datasets. During our study, we verified that it is capable of generalizing well to different image domains (birds, cars and food dishes) and different baseline model architectures of considerably different nature and size (EfficientNet-B0 and CSWin-Large). However, we found that the improvements obtained by using this method were generally not very significant, typically leading to gains of no more than one or two decimal points of accuracy. We also found some cases in which the method did not lead to any improvement at all over the baseline.

During our analysis, we found several aspects of the original FGFR where we believed there was room for improvement. For the specialization step, we found that the original method would result in the specialized heads having a relatively low performance for the 'others' class, which would directly hurt the precision for the rest of the classes of the subset. Regarding the aggregation step, we found it to be notably hard to train. Our main hypothesis is that the per-head classifiers act as a bottleneck during the aggregation step: they are not always right, so the final FC heads cannot confidently learn how to properly combine their predictions.

After this analysis, we came up with a series of improvement proposals with which we extensively experimented as an attempt to improve the fine-grained classification performance of the method. We successfully integrated the uncertainty-based loss weighting method [36] into the optimization of the specialization step, and we proved how it reduced the misclassifications made for the 'others' classes. Also, based on our hypothesis of the per-head classifiers being a bottleneck during the aggregation step, we successfully managed to get rid of them during this step, allowing the final FC layer to directly handle head embeddings, combining them with the mean. This approach, together with the fact that we allowed the complete model to be trained during this step, resulted in substantial improvements with respect to the original method in all the scenarios we considered.

We also demonstrated that our method proposal is still mainly a fine-grained focused method, as the fine-grained classes (those that belong to any subset) are the ones experimenting the largest relative improvement. Nevertheless, we showed that our method is also capable of significantly

improving the performance for the rest of the classes, the generic ones.

Using our method proposal, we achieved a validation performance comparable to some of the state-of-the-art approaches for at least the Food-101 dataset. By replacing the baseline architecture with a more performant one, our results could potentially be further improved. However, this could also imply significantly increasing the complexity of the model. In particular, we find that FGFR is considerably expensive to train, both for its inherent multi-step pipeline (which also introduces the additional cost of having to find suitable hyperparameters for not one but three different steps) and also the evident increase of the model size, typically more than doubling the size of the original baseline model, depending on the number of subsets identified.

We consider it relevant to highlight our findings regarding the lack of performance resulting from using self-attention during the aggregation step. We spent a significant amount of time and effort experimenting with different approaches to make this integration work, as we believed it was a promising strategy, especially after having obtained results such as those shown in Figure 5.18. However, after an extensive experimentation process and comparisons using significance tests, we could do nothing but accept that a much simpler approach (simply taking the head embeddings and computing their mean, without any kind of attention) was undeniably superior in all aspects. Therefore, the idea of using self-attention in this part of the method was discarded.

On the whole, we believe that this work successfully fulfilled its objectives. We managed to replicate and analyze the original FGFR, and we were capable of identifying some of its weak points and propose modifications that tackle them. Our final proposed method showed promising results for a variety of scenarios, and we believe that, with some additional improvements, these results could be further improved while also reducing the computational cost associated to its training process. We discuss some of these improvement ideas in the following section.

6.1 Future Work

We consider multiple research lines that appear to be promising from our current perspective. On the one hand, we believe it could be interesting to verify whether state-of-the-art results could be obtained by applying FGFR+ on top of some other top-performing method for fine-grained classification, or at least other powerful backbone architectures.

We propose continuing with the single-task loss experiments we eventually discarded given temporal and computational restrictions. We believe that the method, namely the specialization step, could highly benefit from using a different loss function, such as focal loss to better deal with its heavy data imabalance, or even some variations of the contrastive loss, so that the heads learn to generate more discriminative vector representations of the fine-grained images.

We also suggest that, in order to fully embrace the potential of multi-task learning, the specialization step could be trained while unfreezing the complete model, rather than just the specialized heads. We hypothesize that, by doing this, the backbone could potentially benefit from the learning of the heads, enhancing the performance of the complete model.

We also consider the possibility of jointly training the specialization and aggregation steps. The goal in this case would be two-fold. On the one hand, by jointly training the specialized heads and

the aggregation part of the network, the whole training of the heads would be oriented towards improving the final classification, rather than just focusing on the per-head performance. On the other hand, combining these two steps into one would significantly reduce the computational costs of training FGFR+, making it more appealing for future use in other applications.

Bibliography

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer Normalization". In: *stat* 1050 (2016), p. 21.

[2] Eustasio del Barrio, Juan Cuesta-Albertos, and Carlos Matrán. "An Optimal Transportation Approach for Assessing Almost Stochastic Order". In: (May 2017).

[3] Ardhendu Behera et al. "Context-Aware Attentional Pooling (CAP) for Fine-Grained Visual Classification". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 2. 2021, pp. 929–937.

[4] Asish Bera et al. "SR-GNN: Spatial Relation-aware Graph Neural Network for Fine-Grained Image Categorization". In: *IEEE Transactions on Image Processing* 31 (2022), pp. 6017–6031.

[5] Lukas Biewald. *Experiment Tracking with Weights and Biases*. 2020. URL: `https://www.wandb.com/`.

[6] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. "Food-101 – Mining Discriminative Components with Random Forests". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 446–461. ISBN: 978-3-319-10599-4.

[7] Antonio Bruno, Davide Moroni, and Massimo Martinelli. "Efficient Adaptive Ensembling for Image Classification". In: *ArXiv* abs/2206.07394 (2022).

[8] Zhao Chen et al. "GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks". In: *International conference on machine learning*. PMLR. 2018, pp. 794–803.

[9] Po-Yung Chou, Cheng-Hung Lin, and Wen-Chung Kao. "A Novel Plug-in Module for Fine-Grained Visual Classification". In: *arXiv preprint arXiv:2202.03822* (2022).

[10] Michael Crawshaw. "Multi-Task Learning with Deep Neural Networks: A Survey". In: *arXiv preprint arXiv:2009.09796* (2020).

[11] Jifeng Dai, Kaiming He, and Jian Sun. "Instance-Aware Semantic Segmentation via Multi-Task Network Cascades". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3150–3158.

[12] Tuong Do et al. "Fine-Grained Visual Classification using Self Assessment Classifier". In: *arXiv preprint arXiv:2205.10529* (2022).

[13] Xiaoyi Dong et al. "CSWin Transformer: a General Vision Transformer Backbone with Cross-Shaped Windows". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12124–12134.

[14] Abhimanyu Dubey et al. "Maximum-Entropy Fine Grained Classification". In: *Advances in neural information processing systems* 31 (2018).

[15] Abhimanyu Dubey et al. “Pairwise Confusion for Fine-Grained Visual Classification”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 70–86.

[16] Jianlong Fu, Heliang Zheng, and Tao Mei. “Look Closer to See Better: Recurrent Attention Convolutional Neural Network for Fine-Grained Image Recognition”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4476–4484. DOI: 10.1109/CVPR.2017.476.

[17] ZongYuan Ge et al. “Fine-Grained Bird Species Recognition via Hierarchical Subset Learning”. In: *2015 IEEE International Conference on Image Processing (ICIP)*. 2015, pp. 561–565. DOI: 10.1109/ICIP.2015.7350861.

[18] ZongYuan Ge et al. “Subset Feature Learning for Fine-Grained Category Classification”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2015, pp. 46–52.

[19] Ross Girshick et al. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[20] Matthieu Guillaumin, Daniel Küttel, and Vittorio Ferrari. “ImageNet Auto-Annotation with Segmentation Propagation”. In: *Int. J. Comput. Vision* 110.3 (Dec. 2014), pp. 328–348. ISSN: 0920-5691. DOI: 10.1007/s11263-014-0713-9. URL: https://doi.org/10.1007/s11263-014-0713-9.

[21] Michelle Guo et al. “Dynamic Task Prioritization for Multitask Learning”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 270–287.

[22] Harald Hanselmann and Hermann Ney. “Fine-Grained Visual Classification with Efficient End-To-End Localization”. In: *arXiv preprint arXiv:2005.05123* (2020).

[23] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[24] Xiangteng He and Yuxin Peng. “Weakly Supervised Learning of Part Selection Model with Spatial Constraints for Fine-Grained Image Classification”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 31.1 (Feb. 2017). DOI: 10.1609/aaai.v31i1.11223. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11223.

[25] Dan Hendrycks et al. “AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty”. In: *arXiv preprint arXiv:1912.02781* (2019).

[26] Joeri R Hermans, Gerasimos Spanakis, and Rico Möckel. “Accumulated Gradient Normalization”. In: *Asian Conference on Machine Learning*. PMLR. 2017, pp. 439–454.

[27] Gao Huang et al. “Densely Connected Convolutional Networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[28] Yanping Huang et al. “GPipe: Efficient Training of Giant Neural Networks Using Pipeline Parallelism”. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[29] Zixuan Huang and Yin Li. “Interpretable and Accurate Fine-Grained Recognition via Region Grouping”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8662–8672.

[30] Sébastien Jean, Orhan Firat, and Melvin Johnson. “Adaptive Scheduling for Multi-Task Learning”. In: *arXiv preprint arXiv:1909.06434* (2019).

[31] Parneet Kaur et al. “FoodX-251: A Dataset for Fine-grained Food Classification”. In: *arXiv preprint arXiv:1907.06167* (2019).

[32] Alex Kendall, Yarin Gal, and Roberto Cipolla. "Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7482–7491.

[33] Masanari Kimura. "Understanding Test-Time Augmentation". In: *Neural Information Processing: 28th International Conference, ICONIP 2021, Sanur, Bali, Indonesia, December 8–12, 2021, Proceedings, Part I*. Sanur, Bali, Indonesia: Springer-Verlag, 2021, pp. 558–569. ISBN: 978-3-030-92184-2. DOI: 10.1007/978-3-030-92185-9_46. URL: https://doi.org/10.1007/978-3-030-92185-9_46.

[34] Jonathan Krause et al. "3D Object Representations for Fine-Grained Categorization". In: *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. Sydney, Australia, 2013.

[35] Jonathan Krause et al. "Fine-Grained Recognition Without Part Annotations". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 5546–5555. DOI: 10.1109/CVPR.2015.7299194.

[36] Lukas Liebel and Marco Körner. "Auxiliary Tasks in Multi-Task Learning". In: *arXiv preprint arXiv:1805.06334* (2018).

[37] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2999–3007.

[38] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. "Bilinear CNN Models for Fine-Grained Visual Recognition". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1449–1457.

[39] Lingqiao Liu, Chunhua Shen, and Anton Van den Hengel. "The Treasure Beneath Convolutional Layers: Cross-Convolutional-Layer Pooling for Image Classification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 4749–4757.

[40] Shikun Liu, Edward Johns, and Andrew J Davison. "End-to-End Multi-Task Learning with Attention". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 1871–1880.

[41] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". In: *arXiv preprint arXiv:1711.05101* (2017).

[42] Ilya Loshchilov and Frank Hutter. "SGDR: Stochastic Gradient Descent with Warm Restarts". In: *arXiv preprint arXiv:1608.03983* (2016).

[43] Jiaqi Ma et al. "Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts". In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 1930–1939.

[44] TorchVision maintainers and contributors. *TorchVision: PyTorch's Computer Vision library*. https://github.com/pytorch/vision. 2016.

[45] Shaobo Min et al. "Multi-Objective Matrix Normalization for Fine-Grained Visual Recognition". In: *IEEE Transactions on Image Processing* 29 (2020), pp. 4996–5009.

[46] Weiqing Min et al. "Large Scale Visual Food Recognition". In: *arXiv preprint arXiv:2103.16107* (2021).

[47] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[48] Yuxin Peng, Xiangteng He, and Junjie Zhao. "Object-Part Attention Model for Fine-Grained Image Classification". In: *IEEE Transactions on Image Processing* 27.3 (2017), pp. 1487–1500.

[49] Jo Plested, Xuyang Shen, and Tom Gedeon. "Non-Binary Deep Transfer Learning for Image Classification". In: *arXiv preprint arXiv:2107.08585* (2021).

[50] Yongming Rao et al. "Counterfactual Attention Learning for Fine-Grained Visual Categorization and Re-identification". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 1025–1034.

[51] Javier Ródenas et al. "Learning Multi-Subset of Classes for Fine-Grained Food Recognition". In: *Proceedings of the 7th International Workshop on Multimedia Assisted Dietary Management*. MADiMa '22. Lisboa, Portugal: Association for Computing Machinery, 2022, pp. 17–26. ISBN: 9781450395021. DOI: 10.1145/3552484.3555754. URL: https://doi.org/10.1145/3552484.3555754.

[52] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International journal of computer vision* 115.3 (2015), pp. 211–252.

[53] Mark Sandler et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[54] Hongbo Sun, Xiangteng He, and Yuxin Peng. "SIM-Trans: Structure Information Modeling Transformer for Fine-grained Visual Categorization". In: *Proceedings of the 30th ACM International Conference on Multimedia*. 2022, pp. 5853–5861.

[55] Mingxing Tan and Quoc Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6105–6114.

[56] Mingxing Tan et al. "Mnasnet: Platform-Aware Neural Architecture Search for Mobile". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2820–2828.

[57] H. Touvron et al. "Grafit: Learning Fine-Grained Image Representations with Coarse Labels". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, Oct. 2021, pp. 854–864. DOI: 10.1109/ICCV48922.2021.00091. URL: https://doi.ieeecomputersociety.org/10.1109/ICCV48922.2021.00091.

[58] L.J.P. van der Maaten and G.E. Hinton. "Visualizing High-Dimensional Data Using t-SNE". English. In: *Journal of Machine Learning Research* 9.nov (2008). Pagination: 27, pp. 2579–2605. ISSN: 1532-4435.

[59] Ashish Vaswani et al. "Attention Is All You Need". In: *Advances in neural information processing systems* 30 (2017).

[60] Catherine Wah et al. "The Caltech-UCSD Birds-200-2011 Dataset". In: (2011).

[61] Yaming Wang, Vlad I Morariu, and Larry S Davis. "Learning a Discriminative Filter Bank within a CNN for Fine-Grained Recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4148–4157.

[62] Xiu-Shen Wei et al. "Fine-Grained Image Analysis with Deep Learning: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[63] Ross Wightman. *PyTorch Image Models*. https://github.com/rwightman/pytorch-image-models. 2019. DOI: 10.5281/zenodo.4414861.

[64] Li Yuan et al. "VOLO: Vision Outlooker for Visual Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[65] Matthew D Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[66] Zhanpeng Zhang et al. "Facial Landmark Detection by Deep Multi-task Learning". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 94–108. ISBN: 978-3-319-10599-4.

[67] Heliang Zheng et al. "Looking for the Devil in the Details: Learning Trilinear Attention Sampling Network for Fine-Grained Image Recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5012–5021.

[68] Zhun Zhong et al. "Random Erasing Data Augmentation". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 07. 2020, pp. 13001–13008.

[69] Haowei Zhu et al. "Dual Cross-Attention Learning for Fine-Grained Visual Categorization and Object Re-Identification". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 4682–4692. DOI: 10.1109/CVPR52688.2022.00465.

[70] Peiqin Zhuang, Yali Wang, and Yu Qiao. "Learning Attentive Pairwise Interaction for Fine-Grained Classification". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 13130–13137.