Fine Grained Image Classification: an exploratory analysis

Borsi Sonia, Leoni Andrea, Mbarki Mohamed

June 4, 2024

Abstract

This paper presents an exploratory analysis of fine-grained image classification, a complex task in computer vision requiring deep learning models to discern subtle differences between highly similar images. The performance of two convolutional neural network (CNN) architectures, ResNet34 and EfficientNetV2, and two transformer-based architectures, ViT16 and SwinT, is compared across three fine-grained datasets: FGVC-Aircraft, Oxford 102 Flower, and CUB-200-2011. The experiments were conducted using pre-trained models that had been fine-tuned on these datasets. The focus was on evaluating the accuracy and loss of each model. EfficientNet consistently demonstrated the highest accuracy, indicating that it is a robust and efficient model for fine-grained classification tasks. SwinT also demonstrated promising results, achieving the highest accuracy on the Mammals dataset used for the Introduction to Machine Learning Competition. Our findings highlight the significance of model architecture and training strategies in attaining high performance in fine-grained visual tasks. The repository of this project is available at: https://github.com/andreleo02/deep-dream-team.git

1 Introduction

This analysis sets out to investigate a particular branch of visual classification, namely fine-grained image classification. The field of fine-grained image analysis (FGIA) [1] is one of the most challenging in computer vision, as it requires the ability of deep learning architectures to recognise and classify images that appear similar but differ in fine-grained features. In general, the classification of images in deep learning can be divided into two main branches: coarse-grained classification and fine-grained classification. Coarse-grained classification is related to the classification of different categories of object that present large differences between them, and thus belong to categories with a large degree of dissimilarity and well-defined boundaries. The second branch is fine-grained classification, which is the focus of this analysis. This is related to the classification of different categories of object that present smaller differences between them, so images of objects with a high degree of similarity. The field of fine-grained image analysis was first introduced to the academic community with the objective of teaching machines to perceive fine-grained details in images. This approach is discussed in detail in the paper "Fine-Grained Image Analysis with Deep Learning: A Survey" (2021) [1], one of the most important ones in the field. Since the first applications of fine-grained classification, it has been evident that this field of computer vision is challenging due to the high degree of similarity between images. In fact, objects belonging to disparate categories or subcategories may exhibit considerable similarity, with minimal inter-class variation. However, they may also exhibit substantial intra-class variation in poses and rotations, rendering them difficult to distinguish. The principal challenge in this domain is the development of algorithms that can effectively identify subtle visual distinctions between objects belonging to the same supercategory or meta-category, but to different subcategories. Indeed, images of the same object captured from disparate angles can exhibit considerable variation. A pertinent example in this specific case could be the differentiation between two models of cars, apparently similar. Furthermore, fine-grained image classification frequently necessitates the input of experts to label the data, which can result in increased costs associated with the experiments conducted within this field of study. Although deep learning has made significant progress in image recognition and classification, for the present task, architectures that have demonstrated excellent performance in the past, such as traditional neural networks, are not sufficiently effective. It is evident that the key to conducting this kind of task is the development of algorithms that are able to accurately

identify the most informative regions in a given image [11].

Figure 1: *Fine grained classification vs general image classification: fine grained classification aims to distinguish between very similar object (red box), while general image classification usually aims to distinguish distinct objects [10]*

In fact, in fine-grained image recognition tasks, the principal deep learning architectures employed belong predominantly to two families: the first family of architectures is based on convolutional neural networks (CNNs), while the second is based on Transformers. In this specific paper, we will be evaluating methods belonging to both families and discussing their differences in terms of architecture and performance in a comparative approach.

2 Related works

2.1 Convolutional Neural Networks (CNNs) for Fine-Grained Classification

Convolutional neural networks (CNNs) have been a fundamental component of numerous approaches in the domain of fine-grained classification in computer vision, due to their robust feature extraction capabilities. The fundamental concept underlying convolutional neural networks (CNNs) is the utilisation of convolution through the incorporation of a convolutional layer into the neural network. This layer performs a dot product between the matrix of learnable parameters (the kernel) and the matrix of perceptive fields. It is not a novel architectural concept; rather, it was first introduced in the late 1980s and early 1990s by Yann LeCun and his colleagues [8]. Their most notable early contribution was the development of the LeNet architecture, which was detailed in a series of papers during that period. Following the release of AlexNet [12], designed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, this architecture gained significant popularity. Subsequently, the model attracted considerable attention and acclaim following its victory in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012, significantly outperforming previous state-of-the-art methods in image classification tasks. In particular, AlexNet [12] and VGG [13], the initial convolutional neural network (CNN) architectures developed by Krizhevsky et al. (2012) and Simonyan & Zisserman (2014) respectively, established the foundation for deep learning in image classification. Since that time, a number of other significant and efficient versions of the standard convolutional neural network have been developed, the most popular of which include GoogleNet [15], ResNet [5], DenseNet [16], HRNet, and EfficientNet [4]. Convolutional neural network architectures are currently the most prevalent black-box structures employed in the field of fine-grained image classification, despite the emergence of another structure, the Transformers, which has demonstrated efficacy in recent years. In this paper we will evaluate the performance of two of the most common and best performing CNN-based architecture: ResNet and EfficientNet.

2.2 Transformer-Based Approaches for Fine-Grained Classification

Although Convolutional Neural Networks (CNNs) are very popular architectures in the field of computer vision, and were the main architectures used to address fine-grained tasks, another important approach has gained significant popularity in recent years. This is the Transformer, which encompasses a range of variants. Transformers, initially designed for natural language processing, have recently demonstrated considerable potential in computer vision tasks, including fine-grained image classification. The fundamental concept underlying the Transformer architecture is that of attention: these networks are designed to identify the most salient elements within an input image, and to extract the features that have been assigned the greatest degree of attention. In more specific terms, self-attention, also known as intra-attention, is an attention mechanism that relates different positions of a single sequence in order to compute a representation of the sequence. Self-attention has been successfully employed in a range of applications, including reading comprehension, abstractive summarisation, textual entailment and learning task-independent sentence representations[9]. The first Transformer was developed by Google and represents the first architecture to rely entirely on attention to compute the representation of its input without the use of classic convolution layers. Attention is a function that connects a query and a set

of key-value pairs to an output, where the query, keys, values, and output are all vectors. By computing the weighted sum of these values, the output is obtained. One of the most significant developments has been the introduction of the Vision Transformer (ViT) [**?**] architecture by Dosovitskiy et al. (2020). This pioneering approach to image classification has been a significant contribution to the field. This was the first instance of applying a Transformer architecture directly to non-overlapping medium-sized image patches. Ovron et al. (2021) proposed the Data-efficient Image Transformer (DeiT) [19], which improves upon ViT by employing strategies such as knowledge distillation to reduce the need for large-scale datasets, thereby making it more practical for fine-grained classification. Swin Transformer, introduced by Liu et al. (2021), is a hierarchical structure with shifted windows: this architectural design is highly effective in modelling both local and global features, and has demonstrated superior performance in fine-grained classification tasks. Other versions of transformers have been developed, including Hybrid CNN-Transformer Models: the Conformer (Peng et al., 2021) integrates convolutional layers for local feature extraction with transformer layers for global context modelling. Additionally, more advanced attention mechanisms have been developed, such as the CrossViT [18], which employs cross-attention between different scales of patches to enhance fine-grained classification. This paper will also focus on two of the most important Transformer-based architectures: the Visual Transformer (ViT) [6] and the Swin Transformer (SwinT) [3].

3 Methods

A previously stated, in order to conduct our experiments on fine-grained image classification, we have selected four models, two belonging to the family of convolutional neural networks - ResNet32 and EfficientNet V2 - and two belonging to the family of transformers - ViT16 and SwinT -, for comparative purposes.

3.1 ResNet

Residual Neural Networks (ResNets) [5] represent a significant advancement in the field of computer vision and deep learning. Having emerged triumphant in the ImageNet competition in 2015, this convolutional neural network demonstrated its efficacy in overcoming a fundamental challenge in deep learning: the capacity to effectively capture features and patterns in complex neural networks. The issue with this kind of task is that, as the network becomes more complex, it becomes increasingly susceptible to the phenomenon of vanishing gradient descent, which makes it challenging to train. In fact, as the number of layers in the network increases and it becomes deeper, the training error will also increase, probably leading to an underfitting of the model. If the added layers can be constructed as identity mappings, a deeper model should have training error no greater than its shallower counter- part [5]. The advent of residual neural networks (ResNets) represented a significant development in the field of deep learning, as they were the first architectures to address the crucial problem of vanishing gradients. The core concept behind ResNets is the introduction of "residual blocks" in the network, which enable the network to focus on learning residuals (the difference between the input and the output) instead of attempting to learn a direct mapping from the input to the output. Formally, if the desired underlying mapping is $H(x)$, ResNet reformulates it as $H(x) = F(x) + x$, where $F(x)$ represents the residual mapping that the network learns2.

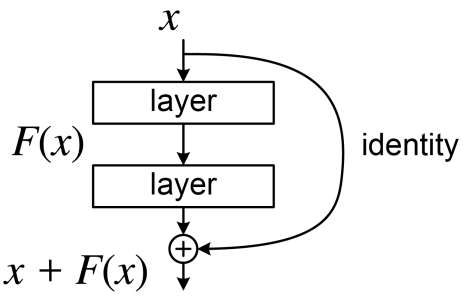

Figure 2: *Residual Block [5] of a Residual Network (ResNet): the Residual Connection skips two layers.*

This reformulation is implemented through the use of "skip connections," another innovation introduced by ResNets. These connections facilitate the flow of the gradient through the network during backpropagation, thereby preventing the vanishing gradient problem. As shown in Figure 2, the typical ResNet architecture consists of various residual block 2, each of them containing a number of convolutional layers, a skip connection that works ad a direct addition of the input x to the output of the convolutional layers, forming $F(x) + x$. A number of variants of ResNet exist, and this paper will consider ResNet34. The number in question indicates the number of layers.

3.2 EfficientNet

As previously discussed, among the various deep learning architectures employed in computer vision tasks, convolutional neural networks have emerged as powerful tools, demonstrating remarkable performance in different tasks. However, as the demand for higher accuracy and efficiency of a model grows, the need for scalable and resource-efficient models becomes increasingly pressing. This is the reason why Tan et al. developed EfficientNet [4]. In their paper, they address the important trade-off of CNNs by proposing a new approach to model scaling. Traditionally, scaling convolutional neural networks architectures was connected to the scaling of one of three dimensions : the width of the model (by adding more feature maps), the depth of the model (adding more layers to the network) or the resolution of the input images. For the first time, with EfficientNet, a new approach has been introduced: compound scaling. This approach focuses on the simultaneous scaling of all three dimensions, rather than on a single dimension alone. Previous scaling approaches have been criticised for their inability to maintain accuracy over time. By scaling just one dimension, initial improvements in accuracy are observed, but this soon reaches a saturation point. In fact, the EfficientNet research team at Google [4] observed that the different scaling dimensions are not independent, which resulted in the necessity to coordinate and balance the different scaling dimensions rather than just one.

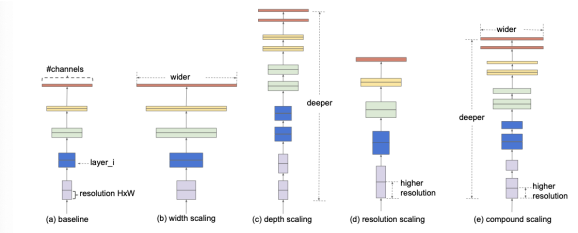

Figure 3: *Model scaling [4]: the figure represents examples of model scaling. The first figure represents the baseline (a), followed by an example of width scaling (b), depth scaling (c) and resolution scaling (d). The last network represents the compound scaling (e), wich scales all the three previous dimensions.*

Inside Efficient Net, three constant coefficients are determined by conducting a small grid search on the model. The depth (d) denoted by α^{Φ}, the width (w) represented by β^{ϕ} and finally the image size (r) represented by γ^{ϕ}. These three values are scaled by a compound coefficient ϕ such that:

$$\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

$$\alpha \geq 1, \quad \beta \geq 1, \quad \gamma \geq 1$$

The objective is to achieve a balance between the three parameters by maintaining a constant ratio. The fundamental concept underlying the compound scaling network is as follows: initially, a small network is constructed, and then a grid search is conducted on the parameters constrained by α, β, and γ. Subsequently, these parameters are fixed as constants and a baseline network is scaled up with different values of ϕ. "Intuitively, ϕ is a user-specified coefficient that controls how many more resources are available for model scaling, while α, β, γ specify how to assign these extra resources to network width, depth, and resolution re- spectively" [4]. The rationale behind EfficientNet is that if the input image is larger, the network requires more layers to expand the receptive field and more channels to capture finer patterns on the larger image. It is possible that superior performance could be achieved by searching for α, β and γ around a large model. However, this would result in a significantly greater search cost. Consequently, this issue is resolved by initiating the search from the outset with the small network. The efficient net is designed to optimise the network for achieving higher accuracy while penalising any computational inefficiency and slow inference times. This is achieved by conducting a single search on the smaller baseline networks and then utilising the same values for the larger models. In the course of our experiments, we will utilise EfficientNetV2 as it represents the optimal choice in terms of both rapid training speed and effective parameter utilisation.[14].

3.3 ViT

The Vision Transformer (ViT) architecture, introduced by Dosovitskiy et al. [6], diverges from the conventional approach of using convolutions to process images. Instead, it splits an image into fixed-size patches, which enables the transformer to capture long-range dependencies, a capability that is particularly beneficial for understanding complex spatial relationships in images. The core idea is to treat image patches as tokens, analogous to words in a sentence, and employ a transformer encoder to model their relationships. The goal of the Vision Transformer developed by the Google research team was to apply a standard transformer directly to images, not only on sentences (NLP), while modifying its architecture as little as possible in order to maintain its original

form. As previously stated, the objective was to divide an image into patches and provide the sequence of linear embeddings of these patches as an input to a Transformer, in order to retain positional information. For position embedding purposes, random z vectors are initialised: these embeddings are learnable throughout the network thanks to backpropagation. A further z-vector is added, designated as the z_0 vector, also known as the learnable class token. This token is also learnable in the same way as the others, with the main difference being that this learnable class token will be the one passed as input to the multi-layer perceptron head, the classifier that will output scores for each category. The other z_n vectors in this stage are not considered, but their output is utilised by intermediate encoders, where each z_i vector is employed as input for the subsequent transformer encoder. The transformer encoder is structured as an alternating sequence of multihead self-attention (MSA) and multi-layer perceptron (MLP) blocks. Prior to each block, layer normalisation (LN) is applied. The MLP also contains two layers with a GELU non-linearity 4:

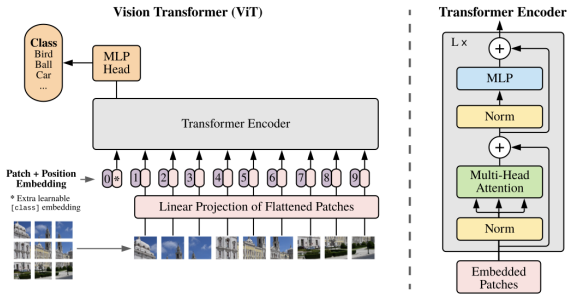

Figure 4: *ViT 16 architecture [6]: as can be seen in this figure, ViT16 splits the input image into fixed-size patches, it linearly embed them and feed this embeddings to the Transformer Encoder. Then, a standard MLP Head is used for classification.*

The architecture of ViT16, as depicted in Figure 4, can be summarized by the following key components. 1. *Patch Embedding*: Each 16x16 patch is flattened and linearly projected to a higher-dimensional space, thereby creating a token for each patch. 2. *The positional encoding*: positional encodings are incorporated into the patch embeddings in order to retain spatial information. 3. *Transformer Encoder*: a series of transformer layers is employed to process the sequence of patch embeddings, with the objective of capturing global context and relationships. 4. *The classification head* is the subsequent component of the model. It predicts the class label based on the final token representations, which have been generated by the transformer encoder.

3.4 SwinT

The Swin Transformer can be considered an improvement on the classical Vision Transformer, which was previously presented. In fact, the SwinT architecture is able to overcome the problem of quadratic complexity, and in fact, it is able to make it even linear while maintaining high accuracy [3]. The Swin Transformer generates a hierarchical representation, commencing with the aggregation of smaller-sized patches and subsequently integrating neighbouring patches in deeper Transformer layers. This hierarchical structure enables the Swin Transformer to utilise sophisticated techniques for dense predictions. The model achieves linear computational complexity by calculating self-attention locally within non-overlapping windows that segment the image. In more detail, the Swin Transformer transforms each pixel of the input image into a patch, with each patch representing a 48-dimensional vector. This is achieved by applying a fully connected layer that performs the linear embedding by mapping each patch into a C-dimensional vector. A noteworthy aspect of the Swin Transformer design is the shifting of the window partition between successive self-attention layers. This shift links the windows from the previous layer, thereby significantly enhancing the model's representational capabilities. In fact, the Swin Transformer block is constructed by replacing the Multi-Head Self-Attention layer, which is typical of standard Transformer blocks, with a shifted-windows based block.

Swin Transformer blocks are computed as:

$$\begin{aligned}
\hat{z}^l &= \text{W-MSA}(\text{LN}(z^{l-1})) + z^{l-1}, \\
z^l &= \text{MLP}(\text{LN}(\hat{z}^l)) + \hat{z}^l, \\
\hat{z}^{l+1} &= \text{SW-MSA}(\text{LN}(z^l)) + z^l, \\
z^{l+1} &= \text{MLP}(\text{LN}(\hat{z}^{l+1})) + \hat{z}^{l+1},
\end{aligned}$$

All other layers remain unchanged. The Swin Transformer block is comprised of two principal blocks, namely W-MSA and SW-MSA. The first component is the W-MSA (window-based multi-scale attention layer): rather than processing the entire image as input, this layer divides it into windows.

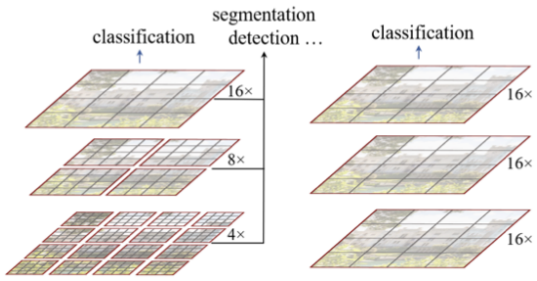

Figure 5: *Shifted Windows (Swin Transformer [3]) vs Normal Windows (Vision Transformer): the Swin transformer on the left (a) builds hierarchical feature maps by merging together the patches of a given image, while normal Vision Transformer (b) produces feature maps of a single row resolution.*

By passing the MSA only a window, rather than the entire set of patches, the complexity is reduced from quadratic to linear. The second important part of the Swin Transformer block is the SW-MSA, which involves shifting each window by half of its size. Each window contains a fixed number of patches, thereby ensuring that the complexity is linear relative to the image size. These features render the Swin Transformer an optimal general-purpose backbone for a range of vision tasks like fine grained image classification, in contrast to earlier Transformer-based architectures, which generate feature maps of a single resolution and have quadratic complexity. The shifted patches are employed to define the windows, which are then passed to a masked MSA (a mask mechanism is defined) and subsequently shifted back to the original patch in a reverse cyclic shift. In order to produce a hierarchical representation, the number of tokens is reduced by means of patch merging layers as the network becomes deeper. Consequently, the shifted-windows approach permits cross-window connections while maintaining the efficient computation of non-overlapping windows. The attention mechanism is computed according to the following formula:

$$\text{Attention}(Q, K, V) = \text{SoftMax}\left(\frac{QK^\top}{\sqrt{d}} + B\right)V$$

where Q is the query matrix, K is the key matrix, V is the value matrix, and B is the bias term. The expression $\frac{QK^\top}{\sqrt{d}} + B$ is passed through the SoftMax function before being multiplied by V. To conduct our experiments we used the SwinT, i.e., the "tiny" version of Swin Transformer having $C = 96$ and *layernumbers* = 2, 2, 6, 2

4 Experimental setup

In this section we will evalute the performance of out four selected models, ResNet34, EfficientNetV2, ViT16, SwinT on our fine-grained image classification task, presenting a few analytical experiments and results.

4.1 Implementation details

In order to assess the efficacy of our models in the domain of fine-grained image recognition, we have elected to utilise the pre-trained models made available by the PyTorch Python library. The training and fine-tuning processes were executed on an Azure machine with GPU (Tesla T4 model) capabilities in order to facilitate the efficient management of the large-scale dataset and our complex model architectures. Given the 150-hour availability on the Azure machine, it was necessary to exercise great care in selecting the computational resources to be allocated to the models. Consequently, we selected versions of each model that offered an optimal balance between complexity and performance, thereby ensuring the most efficient use of the allotted time. Additionally, in consideration of the 150-hour limit, we devised a meticulous experimental plan, comprising sequential phases: the initial phase involved the execution of smaller epochs and less complex models, with the objective of obtaining preliminary outcomes. Thereafter, the complexity of the models was incrementally augmented and also the number of epochs.

4.2 Datasets

To conduct this analysis on fine-grained visual classification, we evaluated the performance of our four models on three very popular datasets in the field of computer vision, specifically chosen for fine-grained tasks like the present one. The first dataset used to train and test our models is the FGVC-Aircraft dataset 6. It contains 10,200 images of aircraft, with 100 images for each of 102 different aircraft model variants. The majority of these variants are airplanes. Aircraft models are organised in a four-level hierarchy.

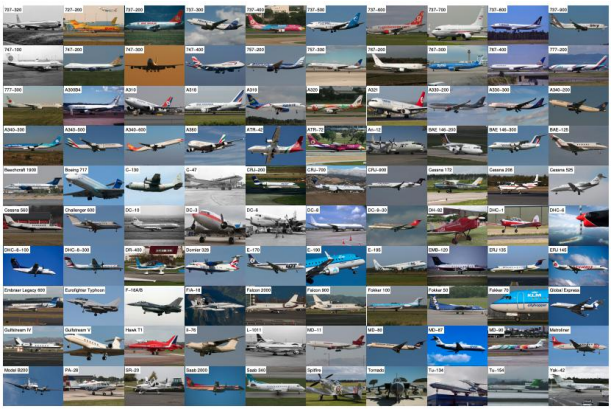

Figure 6: *FGVC-Aircraft dataset: Aircraft, and in particular airplanes, are alternative to objects typically considered for fine-grained categorization such as birds and pets. University of Oxford, https://www.robots.ox.ac.uk/ vgg/data/fgvc-aircraft/*

The second dataset is the Oxford 102 Flower dataset 7. It consists of 102 flower categories. The flowers selected for inclusion in the study are those that are commonly observed in the United Kingdom. Each class comprises between 40 and 258 images.

Figure 7: *Flowers 102 dataset: The images have large scale, pose and light variations. In addition, there are categories that have large variations within the category and several very similar categories. University of Oxford, https://www.robots.ox.ac.uk/ vgg/data/flowers/102/*

Figure 8: *CUB-200-2011 is an extended version of CUB-200. The extended version roughly doubles the number of images per category and adds new part localization annotations. All images are annotated with bounding boxes, part locations, and at- tribute labels. Images and annotations were filtered by multiple users of Mechanical Turk. California Institute of Technology, https://authors.library.caltech.edu/records/cvm3y-5hh21*

The third and final dataset evaluated for the purpose of conducting the experiments is the CUB-200-2011 dataset 8, one of the most widely used dataset for fine-grained visual categorisation tasks. It contains 11,788 images of 200 subcategories belonging to birds, 5,994 for training and 5,794 for testing. It also comprises one subcategory label, 15 part locations, 312 binary attributes and one bounding box.

4.3 Training and Fine-tuning

Transfer learning Each model was initiated with pre-trained weights derived from the ImageNet dataset and already available in Pytorch. The ImageNet dataset is a highly popular large-scale repository of visual data, designed for the specific purpose of facilitating visual object recognition tasks. This is the standard approach underpinning transfer learning, where one leverages a large generic dataset when the number of images is limited for the target task, in our case fine grained image classification[21]. In fact, the utilisation of this approach enabled the leveraging of the extensive feature representations derived from ImageNet, thereby establishing a robust foundation for the models upon fine-tuning them on the specific datasets employed in our experimental setting.

Data preparation The datasets used to train and test the four models in the experiments were split into two subsets: 50% for training and 50% for validation. The aim of this stratified split is to ensure

that both the training and validation subsets of the data are representative of the overall distribution of the data, and this was maintained for all datasets to ensure a fair comparison between the models. In addition, to further improve the generalisation ability of our four models, we applied data transformation techniques to our data, both augmentation and normalisation. More specifically, on the training set we applied a random resized crop, which randomly crops the images to 224x224 pixels. We also set a random horizontal flipping of the images with probability 0.5 and a normalisation using mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225] respectively. These are standard for pre-trained models on ImageNet. For the validation set, we resized the images to 256 pixels, centered them at 224x224, and normalised them as for the test set.

Training process Each model was trained for 50 epochs, with a batch size of 32 and Stochastic Gradient Descent (SGD) employed as the optimizer. The choice of SGD is motivated by its simplicity and effectiveness in training deep learning models. It incorporates momentum (set at 0.9 in our experiments), which facilitates the optimisation process and accelerates convergence. Furthermore, a weight decay factor, also known as L2 regularisation, was incorporated into the models with the objective of imposing a penalty on the loss function, proportional to the sum of the squared values of the model parameters. This approach discourages the model from learning overly complex features, thereby facilitating the maintenance of generalisation to unseen data. Furthermore, an early stopping criterion with a patience of 5 epochs was implemented in order to monitor the performance of the models on the validation phase of the experiment. This approach helped to prevent over-fitting during the training process, thereby ensuring that the computational resources of the machines were conserved. Furthermore, a learning rate of 0.01 was incorporated to further enhance the training process, which is adjusted according to a predefined schedule. The aforementioned schedule is implemented using the Step Learning Rate scheduler from PyTorch. This scheduler systematically reduces the learning rate by a factor *gamma* at regular intervals (step size) during the training of our models. This approach facilitated the model's convergence with greater efficiency. The *gamma* parameter within the scheduler was fixed at 0.1, while the step size was set to 10. Other parameters were commonly shared across the architectures and are the following: number of epochs at 50 and the batch size to 32. These values were found as optimal for all the model's architectures.

Fine-tuning strategy In order to conduct our experiments on the different datasets, we attempted to leverage as much as possible the robust feature representations that had already been learned from ImageNet. The objective was to adapt the models to the Flowers102, CUB and Aircraft datasets with the intention of performing well in the fine-grained image recognition task by fine-tuning. The process entailed three principal stages: layer freezing, hyperparameters adjustment and domain-specific adjustments. With regard to layer freezing, the main goal was to maintain the integrity of the pre-trained feature representation. In particular, we elected to freeze the initial layers of the models, which are responsible for capturing low-level and mid-level features of the images. These features are generally transferable across different image recognition tasks and do not require re-training. With regard to the learning rate adjustment, we proceeded with great care to adjust it during our experiments in order to ensure stable convergence of our model. Initially, the learning rate was set to 0.0001, a relatively low value. However, it was subsequently adjusted and augmented throughout the course of the experiments. A learning rate of 0.01 was employed in two of the models (EfficientNet and SwinT). Instead, for ResNet34 the optimal value found during the experiment was 0.005; ViT16 on the contrary was trained with a significantly higher learning rate of 0.05. Also the weight decay parameter, which perform a L2 regularization, was chosen with respect to the model. In fact, in EfficientNet and ResNet34 a value of 0.001 was used; while in the Transformer-based models (SwinT and ViT16) it was 0.0001. Finally, with respect to our domain-specific adjustments, we attempted to further tailor the models to our specific fine-grained classification task by modifying the output layers of our models. In fact, depending on the different datasets used for each experiment between the three selected, we carefully adapted the final layer of the network to the correct number of classes matching the dataset.

4.4 Metrics

In the course of our experiments, we employed a two different metrics in order to assist us in evaluating the performance of each model on each dataset, both during the training phase and the validation phase: the accuracy and the loss function. The specific loss function employed in our experiments is the Cross Entropy Loss, which is one of the most commonly

used loss functions in the field of visual classification, particularly in fine-grained classification. This function is used to assess the discrepancy between two probability distributions, namely the true distribution of the data and the distribution predicted by the model. Mathematically, it is defined as the negative log-likelihood of the true labels given the predicted probabilities,

$$\text{Loss} = -\sum_{i=1}^{N} y_i \cdot \log(p_i)$$

Where N represents the total number of classes in the given dataset, y_i denotes the true label for class i, and p_i is the predicted probability of class i. With respect to the accuracy, it is a measure of the proportion of correctly classified istances by the model out of the total number.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \times 100$$

During each training epoch, both the cross-entropy loss and the accuracy were calculated on both the training and validation sets. The model weights were saved in correspondence with the optimal set of parameters, which yielded the lowest values of loss and the highest values of accuracy on the validation set.

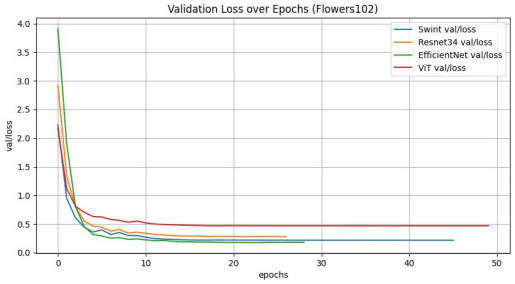

Figure 9: *Validation Loss - Flowers102*

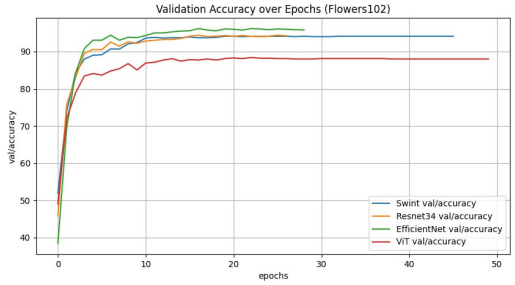

Figure 10: *Validation Accuracy - Flowers102*

5 Findings

5.1 Quantitative results

Evaluation on Flowers102 Dataset The results obtained on the Flowers102 dataset, as shown in Table 1, are noteworthy. The EfficientNet model achieved the highest validation accuracy, at 95.80% 10. This was closely followed by the ResNet34 model, which attained a validation accuracy of 94.22%. The SwinT and ViT16 models also demonstrated high performance, with validation accuracies of 94.12% and 88.01%, respectively. Furthermore, EfficientNet exhibited the lowest validation loss of 0.18 9. Nevertheless, the majority of the models exhibited low validation loss rates: SwinT demonstrated a loss rate of 0.21, followed by ResNet34 at 0.28 and ViT16 at 0.46 .

Evaluation on CUB Dataset Three of the models (EfficientNet, ViT16 and SwinT) tried on this dataset obtained impressive results in terms of accuracy and loss in the validation step 2. In particular, EfficientNet reached an accuracy of 99.91% 12 and a loss of 0.015 11; which normally could be considered unusual results, yet are supported by a training accuracy of 89.73% and a low training loss (0.42). This combination of values suggests that the model is not over-fitting the training data, indeed an optimal combination of *hyper-parameters* were found. As previously stated, the other two models performed in a similar way; respectively, SwinT and ViT16 reached a validation accuracy of 98.18% and 98.36%. ResNet34 performed slightly worse with respect with the other models, however the final results of the architecture are strongly positive (734624% OF ACCURACY!)

Table 1: Flowers102

Model	Train Acc.	Train Loss	Val. Acc.	Val. Loss	Train Time (s)
ResNet34	97.55	0.14	94.22	0.28	168
EfficientNet	97.71	0.11	**95.80**	**0.18**	396
ViT16	97.92	0.13	88.01	0.46	1335
SwinT	98.01	0.07	94.12	0.21	724

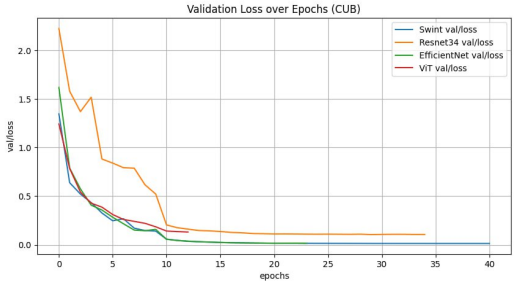

Figure 11: *Validation Loss - CUB*

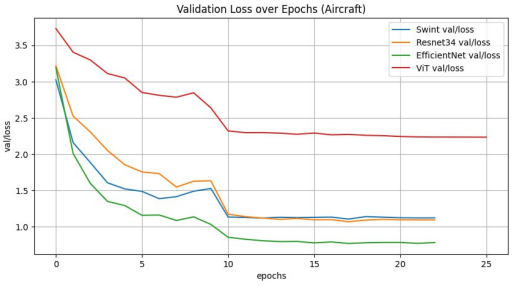

Figure 13: *Validation Loss - FGVCAircraft*

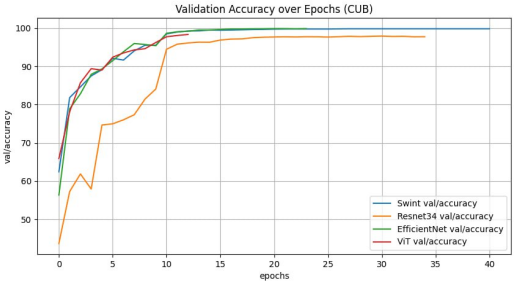

Figure 12: *Validation Accuracy - CUB*

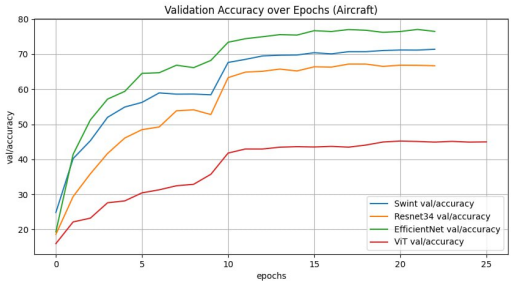

Figure 14: *Validation Accuracy - FGVCAircraft*

Evaluation on FGVCAircraft Dataset In the FGVAircraft dataset 3, the performances are notably lower compared to those of the other datasets. However, EfficientNet is once again demonstrated to be a robust model, exhibiting the highest validation accuracy of 76.49 % 14. This is followed by SwinT with 71.39 % and ResNet34 with 66.69 %. In contrast, the performance of Vit16 is slightly inferior, with a validation accuracy of 44.99 %. In addition, the EfficientNet model exhibits the lowest validation loss of 0.78 13, followed by ResNet34 with 1.09, SwinT with 1.12, and Vit16 with 2.23.

5.2 Qualitative results

The objective of this report was to compare the performances of different state-of-the-art deep learning models in order to evaluate their effectiveness in a fine-grained image classification task. Among the four models tested, it is evident that EfficientNet consistently demonstrated the best performance in terms of accuracy and loss across the different datasets. However, SwinT also showed promising results, indicating the potential of transformers for image classification. Both of these models demonstrate an optimal balance between complexity and efficiency. Indeed, their architectures render them suitable for a diverse range of tasks, including fine-grained classification. With regard to our two Transformers-based models, SwinT exhibited the best performance in compari-

Table 2: CUB

Model	Train Acc.	Train Loss	Val. Acc.	Val. Loss	Train Time (s)
ResNet34	88.22	0.48	97.77	0.10	1046
EfficientNet	89.73	0.42	**99.91**	**0.015**	1610
ViT16	87.37	0.56	98.36	0.13	1558
SwinT	92.27	0.28	98.18	0.24	1180

Table 3: Aircrafts

Model	Train Acc.	Train Loss	Val. Acc.	Val. Loss	Train Time (s)
ResNet34	85.81	0.56	66.69	1.09	555
EfficientNet	89.0	0.4	**76.49**	**0.78**	1185
ViT16	70.63	1.38	44.99	2.23	2025
SwinT	87.17	0.46	71.39	1.12	1115

son to ViT16. The notable difference between the two models is likely due to SwinT's ability to address complex images with its hierarchical approach, which contributes to its robustness in understanding such images. In contrast, ViT16 processes the images in a more "flat" manner, treating each patch equally. This approach may result in a more limited ability to capture fine details and hierarchical information in images, leading to poorer performance. The hierarchical architecture and localised attention mechanisms of SwinT facilitate greater efficiency in terms of computational resources, with a reduction in computational load compared to the global self-attention of ViT16. Indeed, as evidenced by the accompanying tables , SwinT achieves the highest accuracy in the shortest time. With regard to our two CNNs-based models, the discrepancy in performance between EfficientNet and ResNet can be attributed to their respective architectures. In fact, the compound scaling method introduced by EfficientNet has proven to be crucial in terms of model performance. By contrast, ResNet, despite being a deep and effective architecture, does not employ a scaling approach, leading to inefficiencies. The architecture of EfficientNet, with its balanced complexity and efficient use of parameters, is frequently observed to generalise more effectively to unseen data, which is a crucial aspect for fine-grained tasks. The superior performance of all models on the CUB and Flowers102 datasets, and the inferior performance on the FGVAircraft dataset, can be attributed to the complexity of the images. It can be reasonably assumed that the superior overall performance on the CUB and Flowers102 datasets is due to a number of factors, including lower intra-class variability, more distinctive inter-class features, and less complex backgrounds in the former datasets. The FGVAircraft dataset presents more challenging conditions, with higher intra-class variability, subtle inter-class differences, and complex backgrounds (such as airport settings or in-flight scenes). These factors make it more challenging for models to achieve high accuracy and low loss both on validation and on training phases.

5.3 Competition

Dataset For the competition, we trained our models on the Mammalia dataset, which contains 100 different classes of mammals. For the training set, each class of the dataset contained 50 images, giving a total of 5000 images, while the test set contained 10 images per class, giving 1000 test images. To obtain the validation set, 20% of the training set was selected randomly, replicating the same balance of classes present in the training set.

Models We decided to train all the models presented in this report, including those that had already been identified as the best performers in our experiments on fine-grained image classification. In addition, during the competition we had the opportunity to test different versions of these architectures. In particular, we used SwinB [3], which is based on the same Swin base architecture as SwinT but with different size and number of hidden layers (C = 128, layer numbers =2, 2, 18, 2). We also used ResNet50 [5], a more complex version of ResNet (50 layers instead of 32), to see if we could improve its performance on the new dataset.

Strategy The models were selected based on the set of hyperparameters that yielded the optimal

validation accuracy and validation loss outcomes during our previous experiments conducted for this paper. However, given the limited time available for training our models (2 hours), we elected to train each model for fewer epochs. Initially, we trained all our models on just 10 epochs, and then augmented the number of epochs for the models exhibiting the most promising performance.

Results As demonstrated in our previous experiments, the most effective models in the Mammalia dataset were EfficientNet and Swin Transformer as shown in table 4. Nevertheless, in the context of this particular dataset, the Swin Transformer (SwinB) model demonstrated a slight advantage compared to the other models, resulting in the highest accuracy rate and enabling the team to secure third place in the competition.

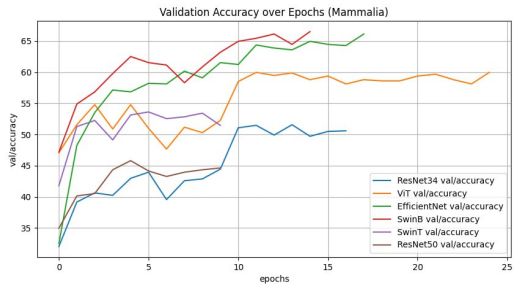

Figure 15: *Validation Accuracy - Mammalia Dataset: SwinB and EfficientNet show the highest accuracy over epochs. SwinT and ViT16 show similar results, while ResNet50 and ResNet34 show the worst performance in terms of validation accuracy.*

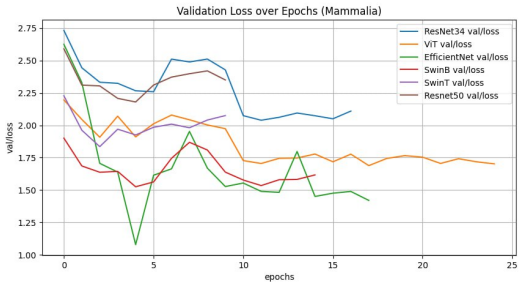

Figure 16: *Validation Loss - Mammalia Dataset: as shown in this plot, EfficientNet shows the lowest level of validation loss over epochs with SwinB. SwinB and ViT16 have similar losses. The highest values correspond to ResNet34 and ResNet50.*

6 Conclusion

The objective of this study was to conduct an exploratory evaluation of four state-of-the-art deep learning architectures—ResNet34, EfficientNetV2, ViT16, and SwinT—on fine-grained image classification tasks using three distinct datasets. Fine-grained image classification is a highly popular field in computer vision and, given the inherent difficulty of this kind of task, we decided to explore the difference in performance between convolutional neural network (CNN) based architectures and transformer-based architectures. Our findings indicate that EfficientNetV2 consistently outperforms other models in terms of accuracy and loss across different datasets, showcasing its capability in effectively handling the nuances of fine-grained classification. SwinT also demonstrated strong performance, particularly due to its hierarchical structure and localized attention mechanisms, making it a viable alternative for such tasks. Conversely, ViT16 exhibited limitations in capturing fine details, resulting in comparatively lower performance. Also, ResNet showed some limitations compared to the other models in terms of accuracy. However, both SwinT and EfficientNet had very promising performances on all the datasets, indicating that both architectures are well suited for fine-grained image classification. During the competition, we attempted to evaluate more complex versions of the Swin Transformer and ResNet to gain further insight. SwinB demonstrated the highest accuracy among all models. Further experiments could be conducted to assess the impact of different model versions and training methodologies on model performance. Our results highlight the critical role of architectural choices and training methodologies in optimizing model performance for fine-grained visual classification. Future work could also investigate further enhancements in transformer-based models and hybrid architectures with the aim of improving their efficacy in fine-grained image analysis.

Notes *This project was carried out by a team of three people: Sonia Borsi, Andrea Leoni and Mohamed Mbarki. We divided the work as follows: Andrea focused mainly on the coding part of the project and did his tests on EfficientNet. Sonia mainly focused on the organization and writing of the paper and conducted tests on Swin Transformer. Both Andrea and Sonia also conducted experiments on ViT16 and organized the GitHub repository. Mohamed concentrated on experiments on ResNet and on the visualisation part of the experiments for the paper. For more information about the code*

Table 4: Mammalia

Model	Train Acc.	Train Loss	Val. Acc.	Val. Loss	Train Time (s)
ResNet34	87.57	0.53	50.58	2.11	312
ResNet50	59.1	1.53	42.48	2.35	191
EfficientNet	88.93	0.44	66.11	1.42	766
ViT16	82.37	0.94	59.96	1.70	1471
SwinT	65.25	1.24	53.61	1.98	351
SwinB	91.05	0.32	66.50	1.61	1916

for this project, please visit our GitHub repository: https://github.com/andreleo02/deep-dream-team.git

References

[1] Xiu-Shen Wei, Member, Yi-Zhe Song, Senior Member, Oisin Mac Aodha, Jianxin Wu, Yuxin Peng, Jinhui Tang, Jian Yang, Member, Serge Belongie. *Fine-Grained Image Analysis with Deep Learning: A Survey.* IEEE TPAMI, 2021.

[2] Po-Yung Chou and Wen-Chung Kao, Fellow. *A Novel Plug-in Module for Fine-Grained Visual Classification.*– Department of Electrical Engineering, National Taiwan Normal University, 2022.

[3] Ze Liu and Yutong Lin and Yue Cao and Han Hu and Yixuan Wei and Zheng Zhang and Stephen Lin and Baining Guo *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows* arXiv, 2021

[4] Mingxing Tan, Quoc V. Le *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks* arXiv:1905.11946v5 [cs.LG] 11 Sep 2020

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun *Deep Residual Learning for Image Recognition* Microsoft Research, 2015

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale* arXiv, 2021

[7] Sangwon Kim, Jaeyeal Nam, Byoung Chul Ko *ViT-NeT: Interpretable Vision Transformers with Neural Tree Decoder*

[8] Yann LeCun, Leon Bottou, Yoshua Bengio, Patrick Haffner *Gradient-based learning applied to document recognition* IEEE, 1998

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Lukasz Kaiser, and Illia Polosukhin *Attention Is All You Need* 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

[10] Li, Zhiwu Lu, Liwei Wang, Tao Xiang, Xinqi Li, and Ji-Rong Wen *Zero-Shot Fine-Grained Classification by DeepFeature Learning with Semantics* arXiv, 2017

[11] Ze Yang, Tiange Luo, Dong Wang, Zhiqiang Hu, Jun Gao, and Liwei Wang *Learning to Navigate for Fine-grained Classification* arXiv, 2018

[12] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton *ImageNet Classification with Deep Convolutional Neural Networks* Part of Advances in Neural Information Processing Systems 25 (NIPS 2012)

[13] Karen Simonyan, Andrew Zisserman *Very Deep Convolutional Networks for Large-Scale Image Recognition* arXiv, 2015

[14] Mingxing Tan, Quoc V. Le *EfficientNetV2: Smaller Models and Faster Training* arXiv, 2021

[15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich *Going Deeper with Convolutions* arXiv, 2014

[16] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger *Densely Connected Convolutional Networks* arXiv, 2018

[17] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, Bin Xiao *Deep High-Resolution Representation Learning for Visual Recognition* arXiv,

[18] Chun-Fu (Richard) Chen, Quanfu Fan, Rameswar Panda *CrossViT: Cross-Attention Multi-Scale Vision Transformer for Image Classification* arXiv, 2021

[19] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, Hervé Jégou *Training data-efficient image transformers and distillation through attention* arXiv, 2020

[20] Hugo Touvron, Matthieu Cord, Matthijs Douze Francisco Massa, Alexandre Sablayrolles, Herve Je gou *Training data-efficient image transformers & distillation through attention* arXiv, 2021

[21] Hugo Touvron, Matthieu Cord, Alaaeldin El-Nouby, Jakob Verbeek, Herve Je gou *Three things everyone should know about Vision Transformers* arXiv, 2022

[22] Andreas Steiner, Alexander Kolesnikov, Xkaohua Zhai, Ross Wightman, Jakob Uszkoreit, Lucas Beyer *How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers* Machine Learning Research, 2022

[23] Ian Goodfellow and Yoshua Bengio and Aaron Courville *Deep Learning* MIT press, 2016

[24] Aurélien Géron *Hands-On Machine Learning with Scikit-Learn & TensorFlow* O'Reilly Media, 2017

[25] Cristopher M. Bishop *Pattern Recognition and Machine Learning* Springer Science and Business Media, 2006

[26] Wah, C. and Branson, S. and Welinder, P. and Perona, P. and Belongie, S. *The Caltech-UCSD Birds-200-2011 Dataset* California Institute of Technology, 2011

[27] Maria-Elena Nilsback, Andrew Zisserman *Automated Flower Classification over a Large Number of Classes* Indian Conference on Computer Vision, Graphics and Image Processing, 2008

[28] S. Maji and J. Kannala and E. Rahtu, M. Blaschko and A. Vedaldi *Fine-Grained Visual Classification of Aircraft* arXiv, 2013