UNIVERSITÀ DI TRENTO

Report Fine-Grained Image Classification

Martina D'Angelo, Enrico Guerriero, Lorenzo Chicco

June 5, 2024, Trento

Contents

1 Introduction

Fine-grained image classification is a task in computer vision that involves identifying different subcategories of a larger class. For example, classifying different species of bird, different species of cars or different species of flowers. This task is considered to be fine-grained because it requires the model to distinguish between subtle differences in visual appearance and patterns. Although fine-grained classification is challenging due to the difficulty of finding discriminative features. Finding those subtle traits that fully characterize the object is not straightforward [16].
In this study, we explore various approaches to address the challenges associated with fine-grained image classification. We scrutinize the application of transfer learning, EfficientNet, DenseNet, SENet, and the ViT Transformer, along with a method for hyperparameter optimization. Despite experimenting with other techniques, such as the NTS-Net (Navigator-Teacher-Scrutinizer Network), we were unable to provide adequate outcomes worthy to be included in the report [17].

Each method offers unique advantages in terms of feature extraction and classification accuracy. Through a series of experiments, we evaluate the performance of these approaches. Our results demonstrate the effectiveness of these methods and provide insights into their strengths and limitations in fine-grained classification tasks.

2 Related Work

Fine-grained image analysis (FGIA) focuses on distinguishing objects within closely related subcategories of a larger category. FGIA deals with objects that are visually similar due to their shared meta-category. This makes FGIA challenging due to small inter-class variations and large intra-class variations in poses, scales, and rotations. The paper "Fine-Grained Image Analysis with Deep Learning: A Survey" [14] by Wei et al. categorizes the existing techniques into distinct paradigms, providing a comprehensive overview of the field's current methodologies:

- *Recognition by Localization-classification subnetworks:* These methods focus on identifying and classifying discriminative regions within the image.
- *Recognition by End-to-end feature encoding:* These approaches aim to learn features directly from the images without explicitly identifying specific regions.
- *Recognition using external information:* These methods incorporate additional data sources, such as textual descriptions or attribute annotations, to improve recognition accuracy.

Among these paradigms, the first two rely solely on the supervisions associated with fine-grained images, such as image labels, bounding boxes, and part annotations. In our paper, we focus on the first two paradigms.
The most commonly used metric in fine-grained recognition is classification accuracy across all subordinate categories of the datasets. It is defined as:

$$Accuracy = \frac{|I_{correct}|}{|I_{total}|}$$

where $|I_{total}|$ represents the number of images across all subcategories in the test set and $|I_{correct}|$ represents the number of images correctly categorized by the model.

Recognition by Localization-Classification Subnetworks These methods involve breaking down the recognition process into two stages: localization and classification. The goal is to identify and focus on the most discriminative parts of the image before classifying them. Techniques within this paradigm include:

- Detection or Segmentation Techniques: These methods use object detection or segmentation to locate key parts of the image, such as bird heads or car lights.
- Deep Filters: Utilizing deep convolutional filters to learn part-level features.
- Attention Mechanisms: Applying attention mechanisms to focus on the most relevant parts of the image.
- Various other techniques, such as multi-agent cooperation or reinforcement learning, to enhance the localization and classification process.

Recognition by End-to-End Feature Encoding. This approach aims to directly learn a unified representation of the image that captures subtle differences between fine-grained categories. It involves:

- High-Order Feature Interactions: Methods that model complex interactions between features to capture fine-grained details.
- Specific Loss Functions: Designing loss functions tailored to improve the recognition of fine-grained features.
- Other Methods: Various additional techniques aimed at improving the discriminative power of the learned representations.

Benefiting from the recent advancements in neural networks, such as VGG [9] and ResNet [5], the ability of these models to extract features has significantly improved. Studies on Fine-Grained Image Classification have evolved from using strongly supervised approaches with additional annotations like bounding boxes to adopting weakly-supervised methods that rely solely on category labels. In these weakly supervised approaches, the focus has primarily been on identifying the most discriminative parts, complementary parts, and parts at various granularities [3].

A study explored the application of Vision Transformers (ViT) in fine-grained classification problems [2]. The results, presented in the paper, demonstrate the significant potential of Vision Transformers for fine-grained visual classification (FGVC). Additionally, their experiments indicate that attention-driven augmentations and the detection of important regions enhance their performance, achieving state-of-the-art results on the Stanford Dogs and CUB-200-2011 datasets. However, the proposed multi-branch, multi-scale architecture has some limitations: the selection of the region of interest based on the attention map is not fully differentiable, preventing the model from being trained end-to-end and necessitating a sequential (multi-stage) training process; ViT-based models demand substantial computational power [2].

3 Methods

3.1 Transfer learning

It is generally not advisable to train a very large Deep Neural Network (DNN) from scratch without first attempting to find an existing neural network that performs a similar task to the one you are tackling [4]. Transfer learning involves taking features learned from one problem and applying them to a new, similar problem [1]. The common approach is to reuse most of the network's layers, except for the top layers. This not only speeds up the training process significantly but also reduces the amount of training data required [4].

For example, if a DNN was trained to classify images into 100 different categories, including animals, plants, vehicles, and everyday objects, and you now need to train a DNN to classify specific types of vehicles, these tasks are similar and even partially overlapping. Therefore, it makes sense to reuse parts of the first network.

The most common incarnation of transfer learning in the context of deep learning is the following workflow [1]:

1. Take layers from a previously trained model.
2. Freeze them, so as to avoid destroying any of the information they contain during future training rounds.
3. Add some new, trainable layers on top of the frozen layers. They will learn to turn the old features into predictions on a new dataset.
4. Train the new layers on your dataset.

If the input pictures for the new task are not the same size as those used in the original task, you typically need to add a preprocessing step to resize them to the size expected by the original model.
More generally, transfer learning is most effective when the inputs share similar low-level features. The output layer of the original model usually needs to be replaced because it is likely not useful for the new task and may not have the correct number of outputs. Similarly, the upper hidden layers of the original model are less likely to be as useful as the lower layers, since the high-level features required for the new task may differ significantly from those needed for the original task. The goal is to find the appropriate number of layers to reuse [1]. The more similar the tasks are, the more layers you will want to reuse (starting with the lower layers). For very similar tasks, try to keep all the hidden layers and just replace the output layer [4].

Finally, an optional step is fine-tuning. It typically starts with lowering the learning rate to ensure that the updates to the model weights are subtle, which helps prevent overfitting. The model, or specific layers, are then trained on a target dataset that closely aligns with the desired task, leveraging the pre-trained knowledge while adapting to the nuances of the new data. This also involves unfreezing all or a part of the previously obtained model and re-train it on the new data with a very low learning rate to achieve meaningful improvements by incrementally adapting the pre-trained features to the new data [1]. This process optimizes the model's performance for specific tasks by refining its ability to generalize from large, diverse datasets to more specialized applications.

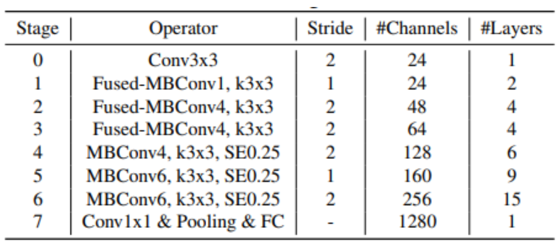
Another option is to train the network using a low learning rate for the pre-trained layers and a higher one for the newly defined one [7].

3.2 Efficient Net

Scaling up convolutional Neural Networks (CNN) is widely used to achieve better accuracy. EfficientNets provides a new compound scaling method that uniformly scales all dimensions of depth/width/resolution. Intuitively, the compound scaling method is logical because with higher resolution input images, the network requires more layers to expand the receptive field and more channels to capture fine-grained patterns. This allows CNN to effectively capture more detailed features in larger images.

In particular EfficientNet achieve an accuracy and efficiency higher than previous CNN while being smaller and faster on inference than the best existing CNN [10]. More recently, the EfficientNetV2 models were created with the aim of improving training speed while maintaining parameter efficiency. In particular, some models of EfficientNetV2, trained much faster on certain experiments than state-of-the-art models while being being up to 6.8x smaller [11]. EfficientNet is a family of models that are optimized for floating point operations per second (FLOPS) and parameter efficiency. The baseline model EfficientNet-B0 is scaled up with a compound scaling strategy to obtain a family of models B1-B7

Figure 4 shows the architecture for EfficientNetV2-S. Compared to the original EfficientNet, this one has several major distinctions, the use of both MBConv [12] and the newly added fused-MBConv in the early layers. Fused-MBConv is a new operation that combines depthwise and pointwise convolutions for faster training and better parameter efficiency. EfficientNetV2 prefers smaller expansion ratio for MBConv since smaller expansion ratio tend to have less memory access overhead. Also, they prefer smaller 3x3 kernel sizes, but it adds more layers to compensate the reduced receptive field resulted from the smaller kernel size. Lastly, they completely removed the last stride-1 stage in the original EfficientNet perhaps due to its large parameter size and memory access overhead.



Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	256	15
7	Conv1x1 & Pooling & FC	-	1280	1

Figure 1: EfficientNetV2-S architecture

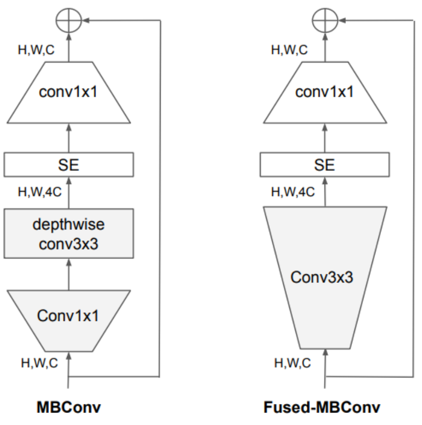


Figure 2: Structure of MBConv and Fused-MBConv

3.3 Dense Net

In the 2016 paper "Densely Connected Convolutional Networks"[6], Huang et al. explain that
"Recent work has shown that convolutional networks can be substantially deeper, more accurate, and efficient to train if they contain shorter connections between layers close to the input and those close to the output - [...] - we embrace this observation and introduce the Dense Convolutional Network (DenseNet), which connects each layer to every other layer in a feed-forward fashion. [...]
DenseNets have several compelling advantages: they alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters."

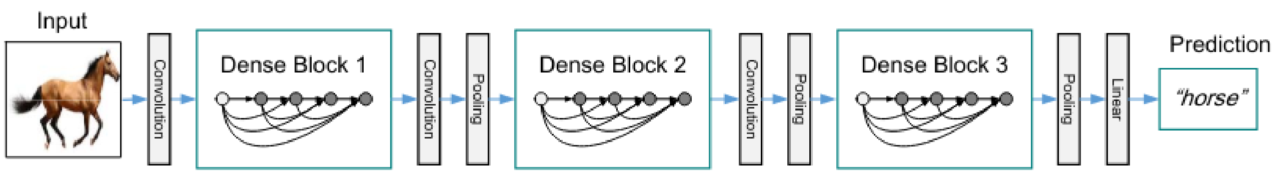


Figure 3: A DenseNet with three dense blocks and four transitional layers[6].

A DenseNet comprises Dense Blocks, containing hidden layers fully connected with each other, and Transition Blocks, performing convolution and pooling. This means that the inputs that each layer will receive will look like this.

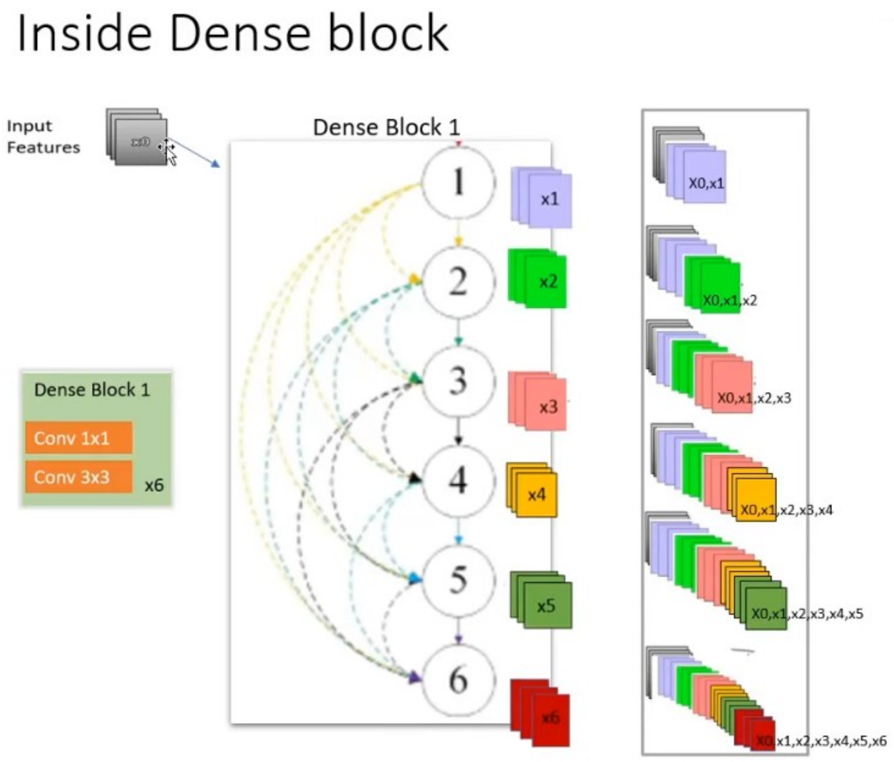


Figure 4: https://www.youtube.com/watch?v=hCg9bolMeJM&t=550s.

In contrast with the previously cited EfficientNet, in a DenseNet the features are never combined together in a summation before passing them onto the next layer. Within a Dense Block, beyond the standard inputs received from its immediate predecessor, each layer also receives concatenated feature maps from all preceding layers as additional inputs. This concatenation ensures comprehensive feature integration across the network.

This characteristic was a key factor in our decision to include a DenseNet in our pool of models for the competition, as Dense Blocks offer several advantages that were particularly beneficial in the competitive scenario:

1. **Alleviate the vanishing gradient problem:** The direct connections from each layer to all subsequent layers facilitate better gradient flow during backpropagation, which is crucial for training deep networks effectively.

2. **Encourage feature reuse:** Each layer has direct access to the feature maps of all previous layers, enhancing the network's ability to leverage existing computations, which is critical in complex pattern recognition.

3. **Less total number of parameters:** By reusing features across layers, DenseNet requires fewer unique parameters compared to traditional architectures, which improves efficiency and reduces the risk of overfitting.

In particular, we chose DenseNet201 (as opposed to DenseNet121, DenseNet169 or DenseNet264) after some empirical trials during the early stages of the project, because it had a good balance between time per epoch and results obtained. This model demonstrated not only high efficiency in learning but also adaptability across various datasets, further validating our choice based on its theoretical merits and practical performance.

The architecture's ability to maintain and utilize the maximum amount of feature information without relearning redundant features makes it particularly suitable for tasks like FGIC.

In DenseNets architectures, the gradient flow is stabilized across the network, which is a significant advantage over traditional deep neural networks. Typically, in deep networks, gradients can diminish (vanish) as they are propagated backward from the output toward the input layers, leading to minimal updates to the weights in the early layers — effectively stalling the learning process in these layers. However, DenseNet's unique structure, significantly counters this issue. Its dense connectivity ensures that the gradient does not significantly lose magnitude as it is propagated back through the network.

All of this means that DenseNets are exceptionally good at learning from a deep set of features without the need for extensive fine-tuning, making them almost plug and play in the context of the competition.

3.4 SENets

A Squeeze-and-Excitation Network (SENet) is a Convolutional Neural Network architecture that employs "squeeze-and-excitation" (SE) blocks to enable the network to perform dynamic channel-wise feature recalibration. The key feature of a SENet is the SE block, which operates in two stages. First, it applies a **Squeeze Operation** that compresses the spatial dimensions of the input feature maps into a single value per channel. This is followed by the **Excitation Operation**, where a small neural network models the interdependencies between channels, producing a set of channel-wise weights or importance scores. The final step is the **Recalibration** of the original feature maps by rescaling them using the learned channel-wise weights [13].

The SE block is beneficial for fine-grained classification as it enhances the network's ability to focus on crucial features within the data. By assigning different weights to channels, the SE block helps the model to learn more effective features from subtle differences in image regions, which is particularly useful for tasks like fine-grained image classification where detailed distinctions are essential. This attention mechanism improves the network's performance by allowing it to adaptively emphasize informative features, leading to better classification results, especially in scenarios with low inter-class variation and high intra-class variation. [15]

3.5 ViT Transformer

A Vision Transformer (ViT) Neural Network is a deep learning model that uses a transformer architecture for computer vision tasks, in contrast to the more commonly used Convolutional Neural Networks [2]. The key aspects of a ViT are:

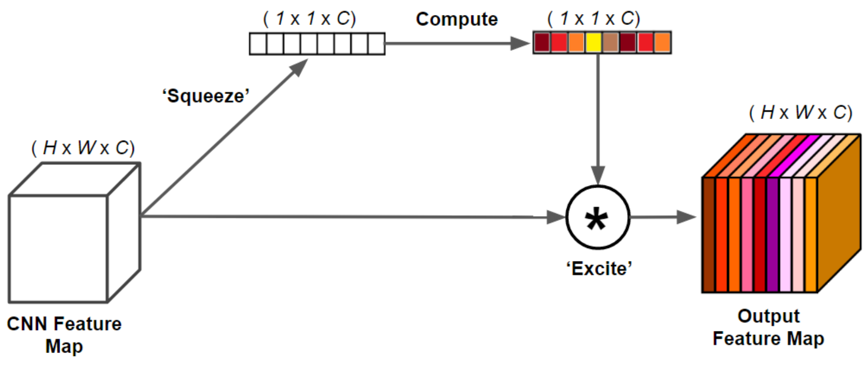


Figure 5: Squeeze and Excitation block

- **Image Patching**: The input image is split into fixed-size patches, which are then linearly embedded into vectors.
- **Positional Encoding**: Positional information is added to the patch embeddings to preserve spatial relationships.
- **Transformer Encoder**: The sequence of patch embeddings is fed into a standard transformer encoder, which uses self-attention to model relationships between patches.
- **Classification Head**: A classification token is added to the sequence, and the final representation of the tokes is used for the final classification.

Traditional CNN-based methods rely on local, low-level features, that may not capture the subtle difference between similar classes in fine-grained classification. ViTs are well-suited for fine-grained classification thanks to their **attention mechanism**: it allows the model to focus on the most informative region of the image [18] [8].

4 Experiments

We wanted to standardize the experiments. We chose the FGVC-Aircraft and Oxford Pets datasets, in order to have two different type of datasets, one smaller and with less classes and the other bigger with more classes, in order to see how the model adapts to different circumstances. Lastly we used the competition dataset, which is unique in its own right since it has less images than Oxford Pets (5000 vs 7400), but has almost the same amount of classes as the FGVC-Aircraft (100 vs 102). This means that the competition dataset is the hardest dataset to perform well on, since it combines a low amount of data with a high number of classes. All the models we used are pre-trained on ImageNet. We experimented with various approaches, such as freezing layers and applying different learning rates to specific layers.

4.1 Experiments with EfficientNetV2-S

Optimizing hyperparameters can pose a significant challenge. A strategy is doing grid search, which involves testing a variety of hyperparameter combinations and assessing their performance. The downside is that this approach is time-consuming.
In our case, we utilized Optuna, a hyperparameter optimization tool, with EfficientNetV2-S. Interestingly, we obtained results that were on par with other methods. However, due to the substantial time investment required by Optuna, the library we integrated to perform the grid search, to determine the optimal hyperparameters, we did not concentrate extensively on this aspect for further analysis.
In our exploration of the Fine-Grained datasets, EfficientNetV2-S demonstrated commendable performance

across the majority of the datasets. However, we encountered difficulties with the competition dataset. This could likely be attributed to our inability to fully optimize the hyperparameters and the limited number of epochs for which we trained the model. EfficientNetV2-S demonstrated superior speed in training, attributable to its compound scaling feature.

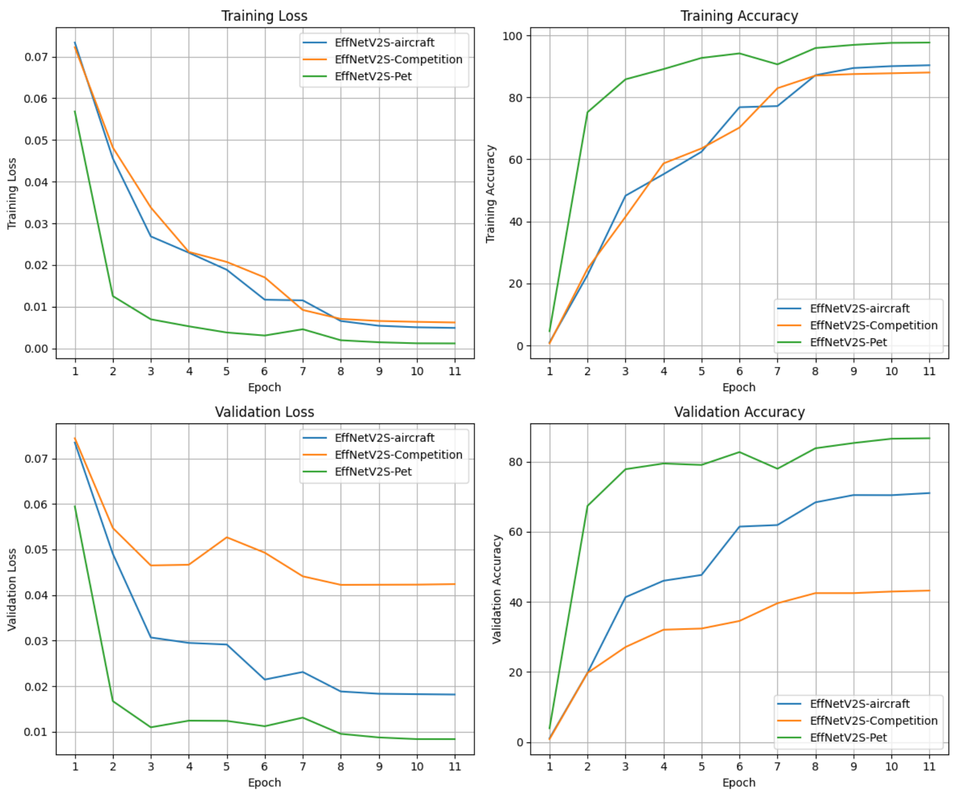


Figure 6: Experiments with EfficientNetV2-S

Model Name	**Dataset**	**Best Epoch**	**Validation Accuracy**
EfficientNetV2S	Aircraft	10	71%
EfficientNetV2S	Competition Dataset	10	43%
EfficientNetV2S	PET	10	87%

Table 1: Model Results

4.2 Experiments with DenseNet

The DenseNet configuration consists of a Batch Size of 64, a Learning Rate of 0.001, a Weight Decay of 0.01, cross-entropy as the loss function, AdamW as the optimizer, StepLR as the scheduler with a step size of 5 and a gamma of 0.5. From the plots, we can see that practically all metrics for every dataset between epochs 5 and 6 immediately degrade when the scheduler comes in. However, after that, we can see that all metrics continue to improve. Between epochs 10 and 11, there is a similar phenomenon although less obvious.

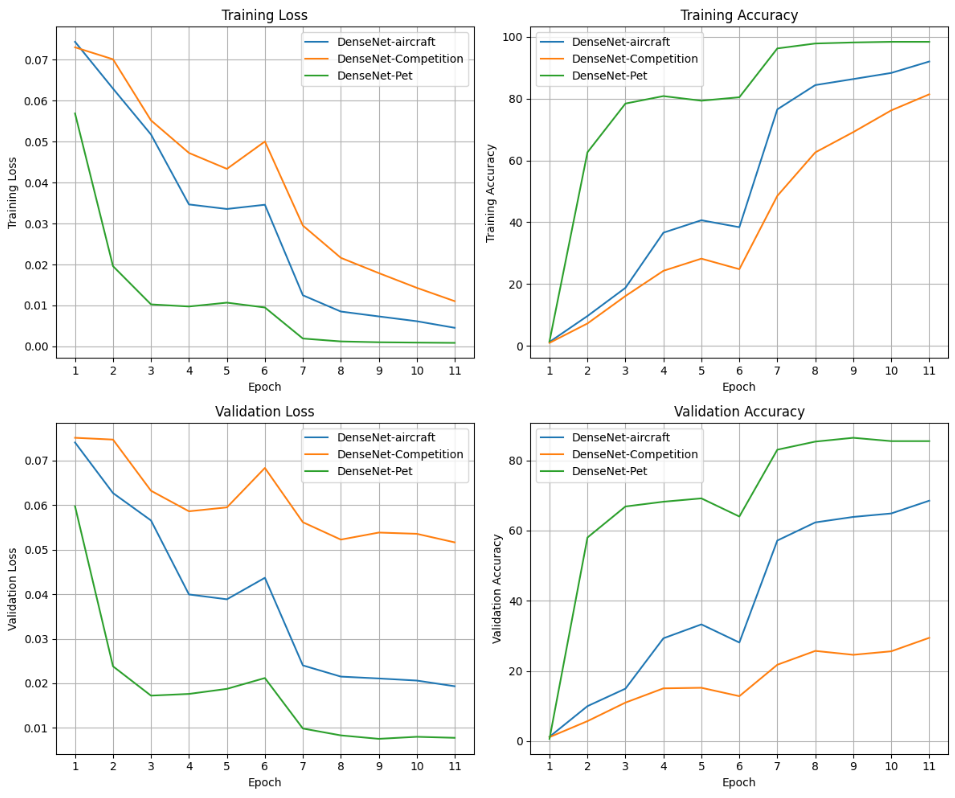


Figure 7: Experiments with Dense Net

Model Name	Dataset	Best Epoch	Validation Accuracy
DenseNet201	Aircraft	10	68%
DenseNet201	Competition Dataset	10	29%
DenseNet201	PET	10	85%

Table 2: Model Results

4.3 Experiments with SEnets

The SENet configuration includes a Scheduler Step Size of 10 and gamma 0.1, a Patience of 5, a Batch Size of 32, a Dropout Rate of 0.5, a Learning Rate of 0.001, a Momentum of 0.9, a Weight Decay of 0.0001, cross-entropy as the loss, AdamW as the optimizer. In particular, for this specific model it is applied a system of freezing and unfreezing that works as follows. The model is instantiated as completely frozen, besides the classifier layer, the last one. After each epoch, one layer from output to input is unfrozen, rendering the model completely unfrozen by the 14th epoch.

Empirically, we noticed that the strategy of freezing allows the model to converge to a higher value of validation accuracy over a greater number of epochs.

We observed that training time is reduced due to freezing, even with an increased number of epochs, which is why this version with gradual unfreezing is considered superior.

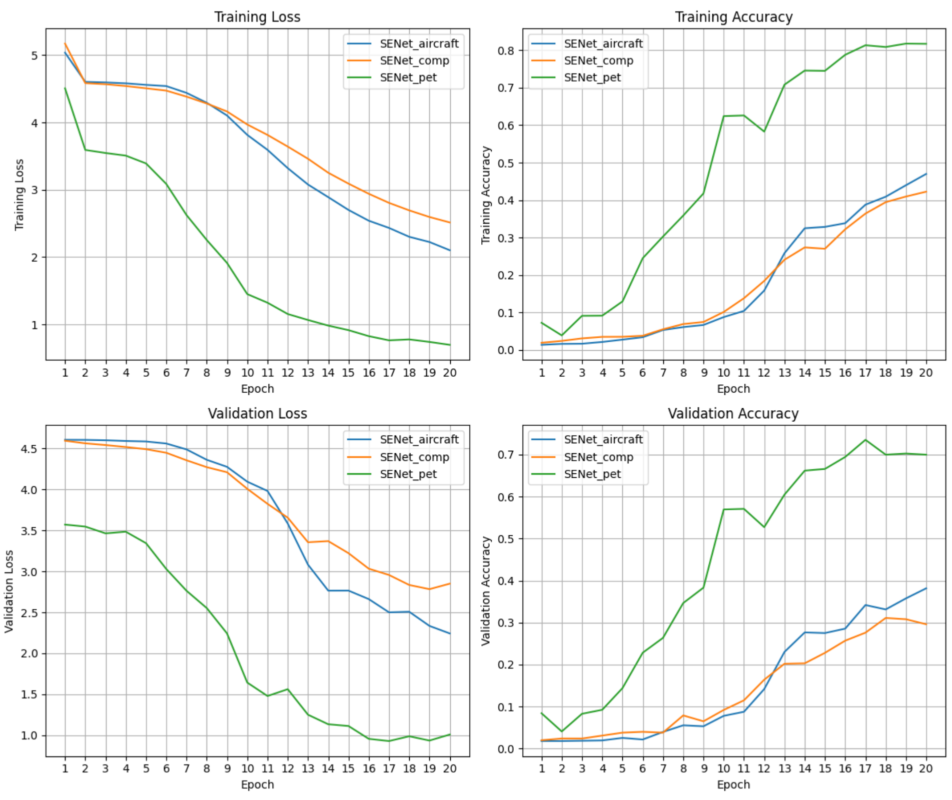


Figure 8: Experiment with SEnets

Model Name	Dataset	Best Epoch	Validation Accuracy
SENet	FGVC-Aircraft	20	38%
SENet	Competition Dataset	18	31%
SENet	PET	17	73%

Table 3: Model Results

4.4 Experiments with ViT Transformer

The ViT Transformer configuration consists of a Scheduler Step Size of 5 and a gamma of 0.1, a Patience of 3, a Batch Size of 32, a Dropout Rate of 0.5, a Learning Rate of 0.001, a Momentum of 0.9, a Weight Decay of 0.0001, cross-entropy as the loss, AdamW as the optimizer. There is a freezing schema applied also to this model, and the reason is that it was prone to overfitting and very slow in training. The final decision is to freeze the first six layers, including the embedding layer and early self-attention layers, to preserve the pre-trained knowledge and enhance generalization on new data.

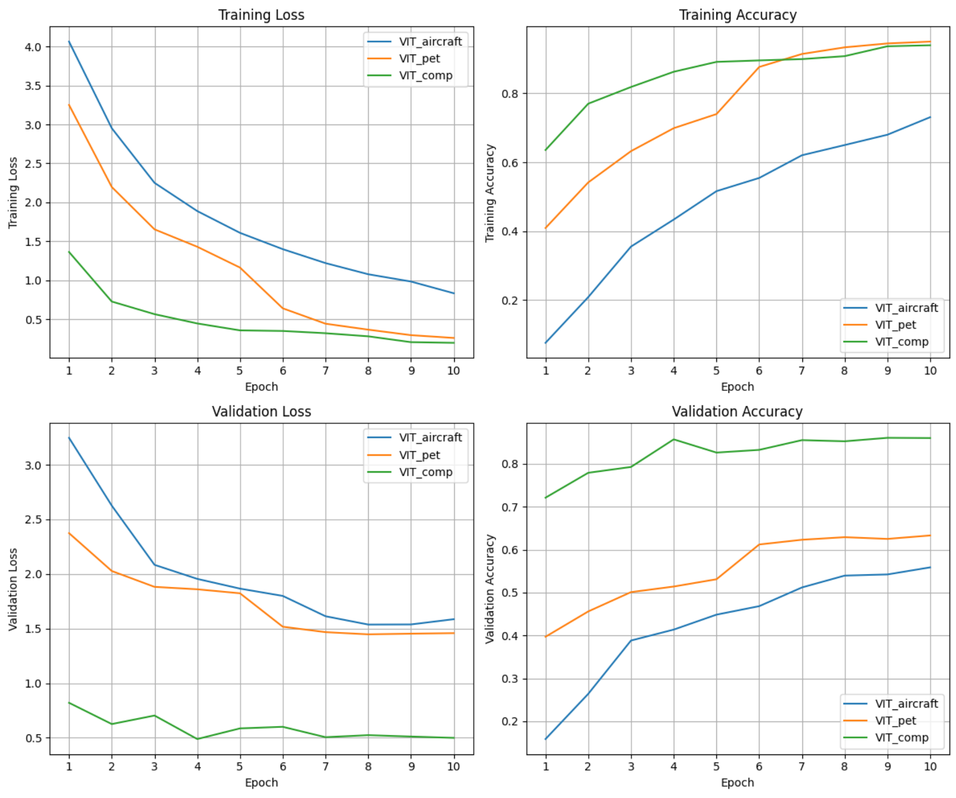


Figure 9: Experiments with ViT Transformer

Model Name	**Dataset**	**Best Epoch**	**Validation Accuracy**
ViT	FGVC-Aircraft	10	75%
ViT	Competition Dataset	10	84 %
ViT	PET	10	86%

Table 4: Model Results

5 Results

In the following plots we can see how the four models performed in the dataset we chose for the experiments.

As mentioned before, the FGVC-Aircraft dataset is a large dataset with many classes, which can tell us how our models perform on a dataset with these two characteristics. In the FGVC-Aircraft dataset we can see how the EfficientNet and ViT models gradually improved their accuracy over time, whereas DenseNet had an important spike in performance after the stepper kicked in. EfficientNet and DenseNet obtained very similar results, whereas ViT got $\approx$ 10% less accuracy. It is important to notice, since it will become more apparent later, that our implementation of ViT tends to perform better than every other model on the very first epoch, while having an overall lower ceiling on its maximal accuracy over many epochs. This hints that our implementation may perform well on small datasets such as the one of the competition, since it can perform well without needing to see much data. We hypothesize that this is due to ViT having been pre-trained on different datasets besides ImageNet, unlike other models. Our implementation of SENet struggled to learn at the same pace of the other models, since it needed more epochs to converge. Lastly, all three best performing models tended to overfit with a difference of around 15% between train and validation accuracy.

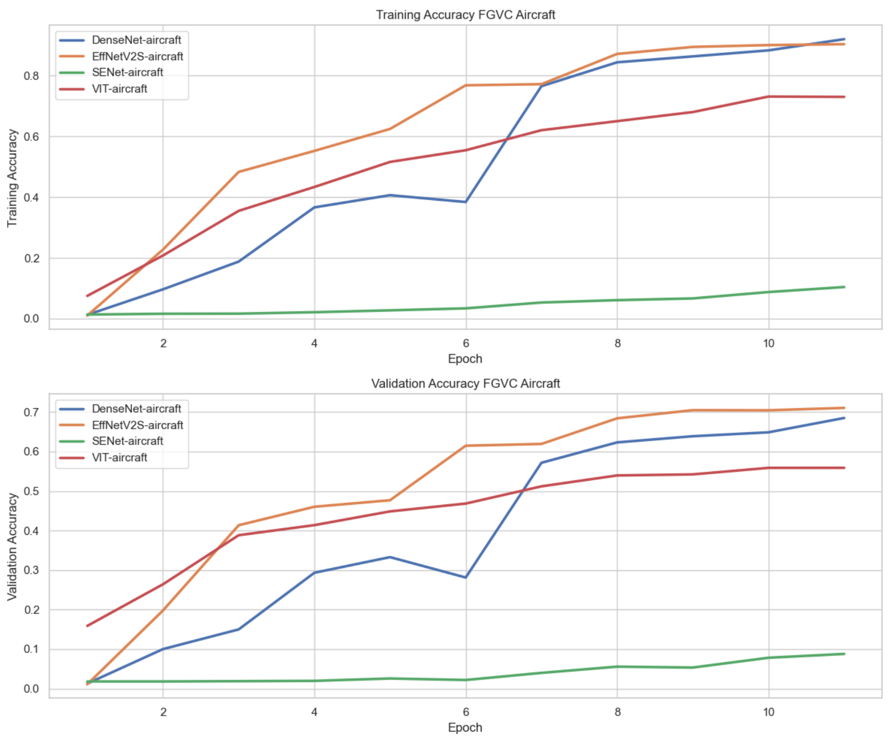


Figure 10: Model Comparisons with FGVC-Aircraft

Next is the Oxford Pets dataset, which is a dataset with fewer classes and images. Due to its dimension, we expected our models to reach excellent results and they did. We can appreciate how ViT is the better model in the first epochs, but once again it tends to converge at a very slightly lower result on training accuracy, but an almost equal result in the validation accuracy, which is the more important metric. By the way, it can be seen that, without unfreezing layers, on small datasets there is the tendency of overfitting (with a training accuracy around 98%). Once again SENet converged more slowly and as shown in its chapter, even with more epochs the results are inferior.

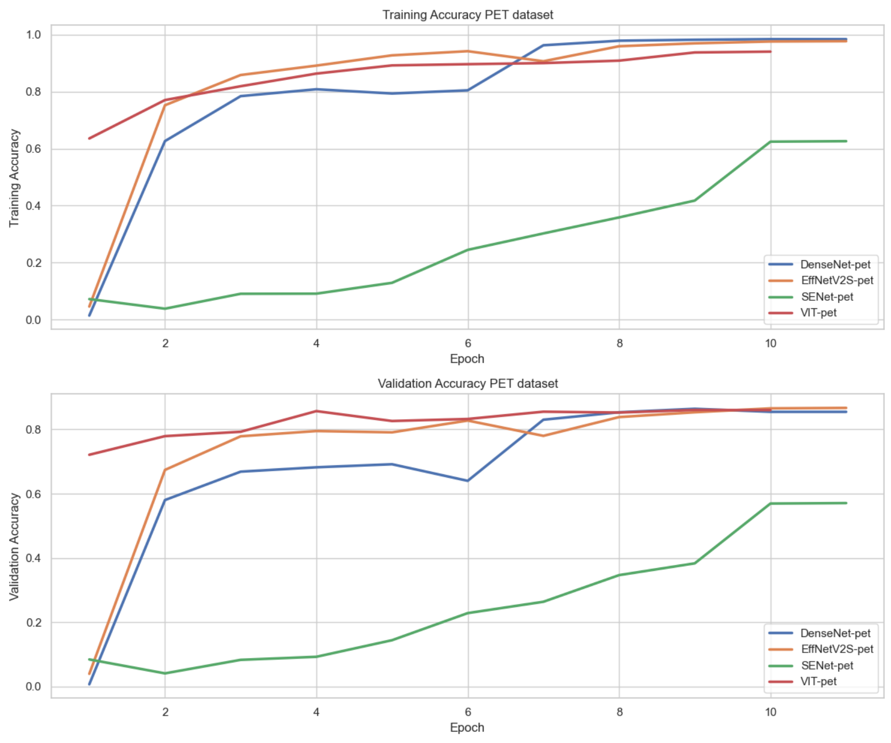


Figure 11: Model Comparisons with PET dataset

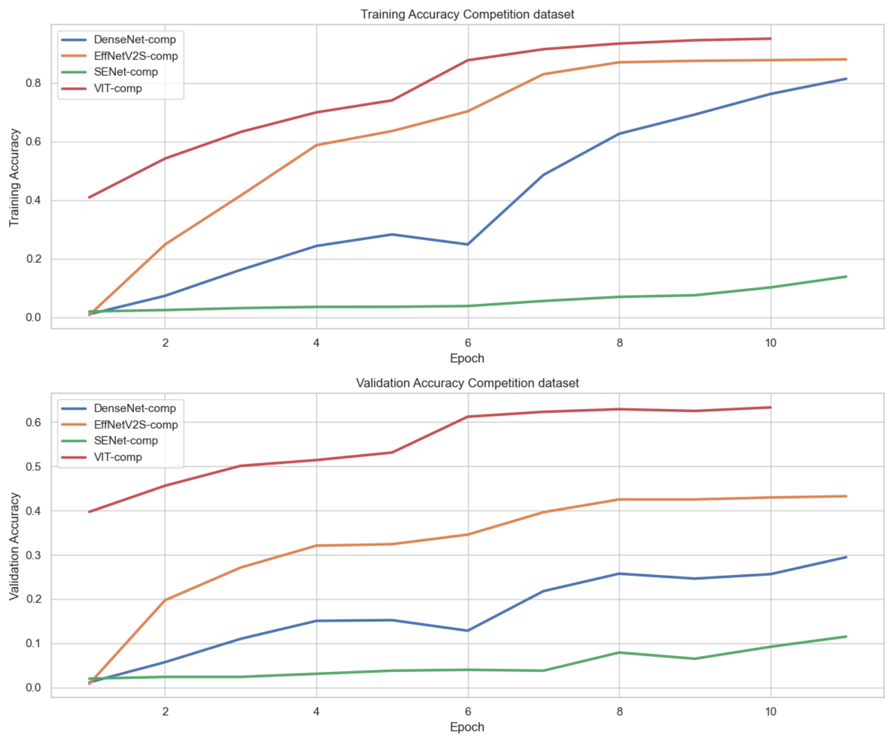


Figure 12: Model Comparisons with Competition dataset

Lastly we have the Competition dataset, which is a dataset with few images and lots of classes. Here is where ViT really shines, outperforming every other model throughout every epoch. The DenseNet started learning really slowly, but after the stepper the slope improved in the training accuracy, but it was very clearly overfitting since the validation accuracy wasn't following the same pace. EfficientNet outperformed DenseNet on both metrics, the model overfit but far less than DenseNet, showing a better generalization.

6 Conclusion

In this study, we have explored various approaches to fine-grained image classification, focusing on the challenges and advantages of different models including SENets, Vision Transformers (ViTs), EfficientNet, and DenseNet. We did experiments on FGVC-Aircraft, Oxford Pets and the competition dataset.

Initially, SENets demonstrated potential, particularly when the number of epochs was increased, as evidenced by the results with the PET dataset at epoch 17. However, their performance exhibited considerable variation across different datasets, underscoring the necessity for dataset-specific adjustments. The performance on the FGVC-Aircraft and competition datasets suggests difficulties in managing more intricate fine-grained distinctions without additional optimization.
ViTs demonstrated good performance across all datasets, particularly excelling in the competition dataset with a 64% validation accuracy. This underscores the effectiveness of ViT in capturing subtle differences in visual features, thanks to their self-attention mechanisms that focus on the most informative regions of the image.

EfficientNet and DenseNet also showcased their strengths in this study. EfficientNet's compound scaling method allowed it to maintain a balance between accuracy and computational efficiency. DenseNet worked well without needing to fine-tune it or change its structure, resulting in solid performance across the datasets tested.

By leveraging pre-trained models and fine-tuning them on specific datasets, we significantly reduced the training time and improved the classification accuracy.

In conclusion, ViT stood out for its superior performance on Oxford Pets and the competition dataset, whereas on FGVC-Aircraft, EfficientNet first, DenseNet second obtained the best results.

7 Notes

Since we learned a lot during this project, there are lots of things that could have been done better.

We tried different approaches other than using pre-made models, like NTS-Net (Navigator-Teacher-Scrutinizer Network) that is a self-supervision mechanism to effectively localize informative regions without the need of bounding-box/part annotations [17] and Few-Shot Learning but unfortunately didn't manage to get good results.

We wasted a lot of time writing from scratch functions for data augmentation or organization that are already present (and most definitely better implemented) in libraries such as PyTorch, Scikit-Learn, TensorFlow etc.

We should have probably opted for a structure that used classes instead of functions to train the models. We learned about the possibility of using datasets present in PyTorch and libraries such as Optuna only during the last few days. In fact we had problems working with datasets such as FGVC-Aircraft up until the last moment. We wrote functions that mapped the images to their correct class as folders manually, but at the end of the day they didn't work, since we ended up with 70 classes instead of 102 (the results we have shown previously were computed on the correct dataset). Regarding Optuna, only one member (Martina) of the team managed to make it work and understand how to use it to find the optimal hyperparameters, the rest manually tried different routes to optimize the models. The GitHub wasn't managed in the best way, we all worked on it but probably had different ideas on the implementation (despite agreeing on the end goal, we just thought about using different routes) and when somebody made an edit, something broke for the other members. However, a lot of these issues were probably unavoidable since we didn't know enough about working as a team managing a common repository. We had to fail in order to learn some important lessons about how to collaborate on a project like this one in the future.

Lastly, due to unfamiliarity with a task like FGIC, we couldn't make an educated guess prior to testing in first person different models. All the information we gathered was through papers, but we weren't able to implement the majority of them from scratch. This is also the reason why we mainly focused on choosing functional architectures rather than something more specific for the specific task at hand.

7.1 Repository

GitHub repository: https://github.com/lorenzochicco99/intromlproject

For this project, our group prioritized making the repository functional and comprehensible for everyone involved. We started by establishing a clear structure, with Martina and Lorenzo taking the lead on setting it up. Meanwhile, Enrico reviewed relevant papers and tested the code in his notebooks.
Once the repository was operational, Lorenzo continued to maintain it while Martina and Enrico focused on reading papers and experiment with different methods and models using sample datasets. Each member focused on a specific model: Lorenzo worked on DenseNet, Martina on EfficientNet, and Enrico on ViT and SENets.
Everyone contributed equally to the creation and optimization of the repository and their chosen models.

8 Bibliography

References

[1] François Chollet. *Transfer Learning and fine-tuning*. https://keras.io/guides/transfer_learning/. Accessed: 2024-05-20.

[2] Marcos V Conde and Kerem Turgutlu. "Exploring vision transformers for fine-grained classification". In: *arXiv preprint arXiv:2106.10587* (2021).

[3] Ruoyi Du et al. *Fine-Grained Visual Classification via Progressive Multi-Granularity Training of Jigsaw Patches*. 2020. arXiv: 2003.03836 [cs.CV].

[4] Aurelien Geron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2nd. O'Reilly Media, Inc., 2019. ISBN: 1492032646.

[5] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[6] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks". In: *CoRR* abs/1608.06993 (2016). arXiv: 1608.06993. URL: http://arxiv.org/abs/1608.06993.

[7] Elisa Ricci. "Deep Learning Lab UniTn Course".

[8] Guangzhe Si et al. "Token-Selective Vision Transformer for fine-grained image recognition of marine organisms". In: *Frontiers in Marine Science* 10 (2023). ISSN: 2296-7745. DOI: 10.3389/fmars.2023.1174347. URL: https://www.frontiersin.org/articles/10.3389/fmars.2023.1174347.

[9] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[10] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *CoRR* abs/1905.11946 (2019). arXiv: 1905.11946. URL: http://arxiv.org/abs/1905.11946.

[11] Mingxing Tan and Quoc V. Le. “EfficientNetV2: Smaller Models and Faster Training”. In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 10096–10106. URL: http://proceedings.mlr.press/v139/tan21a.html.

[12] Mingxing Tan and Quoc V. Le. “MixConv: Mixed Depthwise Convolutional Kernels”. In: *CoRR* abs/1907.09595 (2019). arXiv: 1907.09595. URL: http://arxiv.org/abs/1907.09595.

[13] Fubo Wang et al. “A Fine-grained Classification Method of Thangka Image Based on SENet”. In: *2022 International Conference on Cyberworlds (CW)*. 2022, pp. 23–30. DOI: 10.1109/CW55638.2022.00013.

[14] Xiu-Shen Wei et al. “Fine-Grained Image Analysis with Deep Learning: A Survey”. In: *CoRR* abs/2111.06119 (2021). arXiv: 2111.06119. URL: https://arxiv.org/abs/2111.06119.

[15] Tee Yee Yang. “Introduction to squeeze excitation networks”. In: *Medium* (2020). URL: https://towardsdatascien
com/introduction-to-squeeze-excitation-networks-f22ce3a43348.

[16] Ze Yang et al. “Learning to Navigate for Fine-grained Classification”. In: *CoRR* abs/1809.00287 (2018). arXiv: 1809.00287. URL: http://arxiv.org/abs/1809.00287.

[17] Ze Yang et al. “Learning to Navigate for Fine-grained Classification”. In: *CoRR* abs/1809.00287 (2018). arXiv: 1809.00287. URL: http://arxiv.org/abs/1809.00287.

[18] Xiaohan Yu et al. “Mix-ViT: Mixing attentive vision transformer for ultra-fine-grained visual categorization”. In: *Pattern Recognition* 135 (2023), p. 109131. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2022.109131. URL: https://www.sciencedirect.com/science/article/pii/S0031320322006112.