A Comparison of Deep and Transfer Learning Approaches for Galaxy Morphology Classification from Images

Riccardo Alberghi, Edoardo Calderoni, Matteo Casiraghi, Mattia Forzo

May, 2024

Abstract

Astronomy generates vast amounts of image data, which necessitate reliable automated classification algorithms that can capture the subtle differences in galaxy shapes. We explore a wide range of supervised strategies, from fine-tuning pre-trained models to devising our custom architecture. Finally, we approach the problem in an unsupervised fashion, leveraging DINO embeddings and clustering techniques. Through comparative analysis, we identified the most effective methodologies for the classification of galaxy morphology, providing insights into the broader question of model development versus adaptation in computer vision.

1. Introduction

When tasked with solving a problem using a neural network, we encounter a critical decision point: should we build a model from scratch, or should we leverage pre-existing models and adapt them to our specific task? This report aims to provide a comprehensive analysis to answer this fundamental question.

The task at hand revolves around one of our shared passions: astronomy. To grasp the scale of data production in this field, consider that the Square Kilometer Array (SKA) radio telescope project is anticipated to generate over an exabyte of raw data daily. Our objective is to automate the classification of galaxy morphology, a task that has been historically performed manually, but which is now increasingly reliant on automatic algorithms. This task's complexity is heightened by the often subtle differences between the morphologies of various galaxy types, making it a non-trivial challenge.

In our experiments, we evaluated several models and training procedures to address our research question. Specifically, we compared the following approaches:

1. A VGG architecture trained from scratch, without any pretraining.
2. A VGG11 architecture pretrained on ImageNet-1k, utilizing both shallow and deep fine-tuning techniques.
3. A custom convolutional neural network developed from scratch, that leverages early average pooling for noise reduction, robustness and performance.
4. A pretrained DINO model, where the CLS token is used for k-means clustering to perform completely unsupervised training.
5. A fine-tuned DINO model on galaxy morphology data, where the CLS token is used for k-means clustering to perform completely unsupervised training.

By comparing these different methodologies, we aim to determine the most effective approach for the automatic classification of galaxy morphology.

2. Data

For our task, we used the Galaxy10 DECals Dataset, which contains 17736 colored images of galaxies, in 256×256 resolution.

The labels come from Galaxy Zoo, a crowdsourced astronomy platform where volunteers are invited to classify a large number of galaxy images. Galaxies are divided into ten classes according to their shape, as seen in Figure 1:

1. Disturbed Galaxies
2. Merging Galaxies
3. Round Smooth Galaxies
4. In-between Round Smooth Galaxies
5. Cigar Shaped Smooth Galaxies
6. Barred Spiral Galaxies
7. Unbarred Tight Spiral Galaxies
8. Unbarred Loose Spiral Galaxies
9. Edge-on Galaxies without Bulge
10. Edge-on Galaxies with Bulge

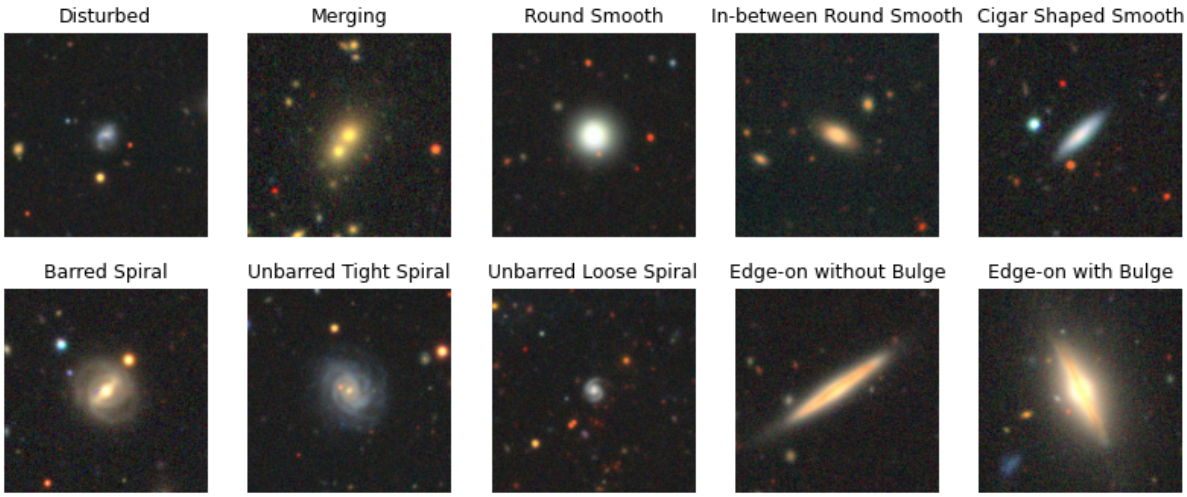

Figure 1: An example from each of the ten classes in the dataset

3. Related work

3.1. VGG11

The VGG11 architecture, introduced by Simonyan and Zisserman in 2015, is a convolutional neural network (CNN) designed for large-scale image classification tasks. It is part of the Visual Geometry Group (VGG) series of models, which are known for their simplicity and effectiveness. The architecture consists of 11 layers: 8 convolutional layers and 3 fully connected layers.

The convolutional layers use small receptive fields of 3x3 pixels with a stride of 1 and padding of 1, maintaining the spatial resolution of the input. Max-pooling layers, with a 2x2 pixel window and a stride of 2, follow certain convolutional layers to reduce the spatial dimensions and control overfitting. The convolutional layers are organized into five blocks, each followed by a max-pooling layer. The final part of the VGG11 model consists of three fully connected layers, with the first two having 4096 neurons each, and the final layer containing 1000 neurons for classification, corresponding to the number of classes in the ImageNet dataset. The network employs ReLU activation functions after each convolutional and fully connected layer, and a softmax activation function at the output layer to produce probability distributions for class predictions.

VGG11 is renowned for its depth and simplicity, providing a good balance between computational efficiency and accuracy. It has been widely used as a backbone for various computer vision tasks and serves as a strong baseline in transfer learning applications, making it a relevant model for the automatic classification of galaxy morphology.

3.2. DINO

DINOv1 (Distillation with No Labels) is a self-supervised learning framework introduced by Caron et al. in 2021. It leverages Vision Transformers (ViTs) and aims to learn meaningful representations of images without requiring labeled data. The architecture's name reflects its core approach of using knowledge distillation in a self-supervised manner.

In DINOv1, two networks are used: a student network and a teacher network, both typically implemented as Vision Transformers. The teacher network generates pseudo-labels from the input data, which the student network then learns to predict. Unlike traditional supervised learning, DINOv1 does not rely on manually annotated labels but instead uses the teacher's output as targets.

The Vision Transformer (ViT) architecture within DINOv1 treats an image as a sequence of patches, similar to the way words are treated in natural language processing models. An image is divided into fixed-size patches, which are then linearly embedded and combined with positional embeddings. These embeddings are fed into a standard transformer encoder comprising multiple layers of multi-head self-attention and feed-forward neural networks.

A unique aspect of DINOv1 is the use of a CLS (classification) token, which aggregates information from all patches and is used for downstream tasks. During the training phase, different augmentations of the same image are fed into the stu-

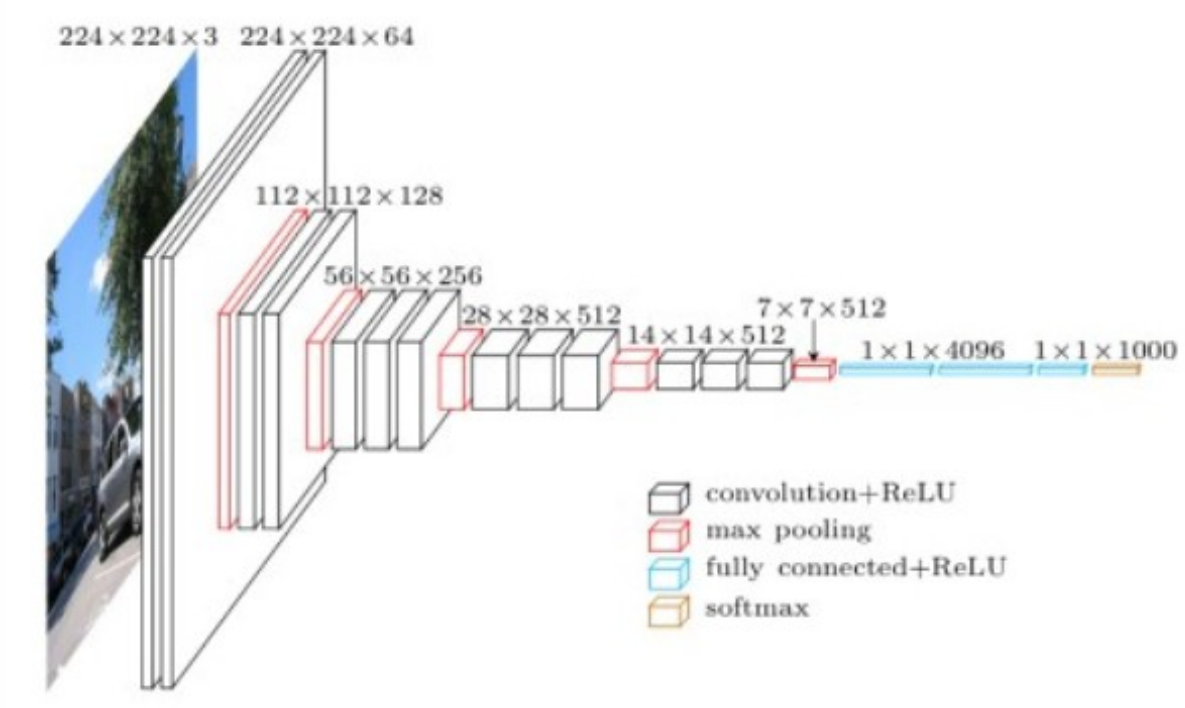

Figure 2: The VGG11 architecture, as described in the original paper

dent and teacher networks. The student network learns to align its CLS token representation with that of the teacher network using a cross-entropy loss, effectively distilling knowledge from the teacher to the student.

DINOv1 demonstrates strong performance in unsupervised representation learning, producing embeddings that are competitive with supervised counterparts on various benchmarks. It has been applied successfully in tasks such as image classification, clustering, and retrieval, making it a pertinent model for our exploration of unsupervised galaxy morphology classification.

4. Methods overview

In this section, we will present in more detail how the various approaches are implemented.

For the sake of consistency, we used the same hyperparameters to train each different version. AdamW has been used with a cosine annealing learning rate scheduler starting from a learning rate of 5×10^4 reaching a minimum of 1×10^{-6} at the last epoch. Each model has been trained for 100 epochs with a batch size of 64.

4.1. VGG trained from scratch

We used the VGG class from the torchvision package, but decided to customize the architecture by setting our own configuration of convolutional layers, number of channels and max pooling.

In the original VGG paper, the number of channels kept increasing in the deeper layers of the network, starting from 64 and going up to 512. We opted for a different solution, as galaxy morphology classification does not require as many high level features as the ones needed in large and varied classification tasks such as ImageNet-1k. However, we still need many low-level features, as those need to capture the basic 3x3 kernels that one can encounter. We ultimately decided to keep the number of channels constant to 32 throughout the network.

As in the original VGG paper, weights are initialized randomly following a normal distribution with mean 0 and variance 10^{-2}.

4.2. VGG11 pretrained with deep fine-tuning

As a starting point, the VGG11 architecture pretrained on ImageNet-1k is taken. The goal of this approach is to retain the low level features learned in pretraining, freezing the first convolutional layers and substituting the later layers, as well as the final classifier head. In doing this, we expect the low level features to get recombined in a more meaningful way to solve the galaxy morphology classification task.

Figure 3: Examples of attention maps from DINO

4.3. VGG pretrained with shallow fine-tuning

The approach is very similar to point 3.2, except for the fact that, here, we only substitute the head, freezing both low and high level features from the convolutional part of the network. Only the classifier head is reinitialized and trained.

4.4. Custom architecture

Our custom architecture is based on the assumption that the high-level features that are useful to determine a galaxy morphology are fewer than what is needed in ImageNet classifiers. For this reason, as in 3.1, we kept the channels constant to 32, but made the network slightly shorter.

The key element in our architecture is an initial average pooling layer that is convenient for two reasons. Firstly, it reduces the natural occuring noise due to the image detector. Secondly, it allows us to have a shorter network without running into issues regarding the receptive field and information transmission.

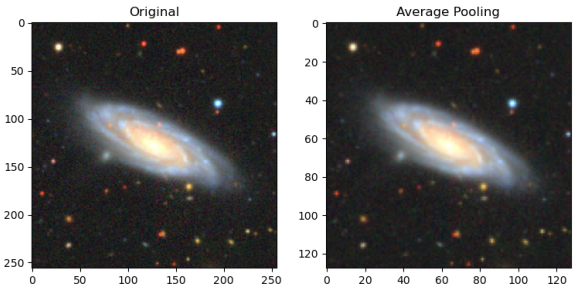

Figure 4: A scheme

4.5. Unsupervised classification with DINO

As one last methodology, we wanted to check whether training was actually necessary to solve this task, or if it can be solved by directly using a pretrained model with the k-means clustering algorithm taking the role of a classification head.

DINOv1, with architecture ViT small patch size 16, pretrained on ImageNet-1k, was used to generate embeddings for the entire dataset of galaxies. The embedding is the concatenation of the CLS tokens extracted from the last 4 encoder blocks of DINO. Then the embedding of the entire dataset are fed into a PCA that retains 95% of the variance lowering the dimensionality from 1536 to around 100. Finally clustering with k-means is performed using Hungarian matching.

To adapt the model to this very specific task, we have performed a fine-tuning of DINO on the Galaxy10 dataset. To reduce workload, the student model was trained with Low Rank Adaption (LoRA).

5. Results

Benchmark		
Model	Accuracy	Roc Auc
VGG11 from scratch	73%	0.924
VGG11 deep tuning	74%	0.924
VGG11 shallow tuning	58%	0.880
Custom architecture	76%	0.929
Dino + k-means	16%	0.521
Dino + k-means, tuned	16%	0.522

As illustrated in the table, comparing the VGG methodologies, the model that achieved the best results was the VGG architecture with deep fine-tuning. In contrast, the VGG model with shallow fine-tuning exhibited a significant drop in accuracy. This discrepancy is primarily due to the divergence between the high-level features used for classifying common objects in ImageNet-1k and the unique structure of galaxies.

From these observations, we can conclude that when

the data distribution for fine-tuning differs substantially from the pretraining data, it is more effective to either retrain the network completely or perform deep fine-tuning. This ensures the model can adapt to the specific characteristics of the new dataset, thereby improving classification performance.

Our custom architecture outperforms all VGG methodologies in our experiments. Notably, this architecture provides a significant advantage in terms of flexibility. Developers can tailor the model to achieve an optimal balance between speed and accuracy based on specific requirements. This adaptability ensures that the custom architecture can be modified to meet specific project goals, whether the priority is achieving the highest possible accuracy or optimizing for computational speed. This balance of performance and flexibility makes our custom model a robust solution for the automatic classification of galaxy morphology.

Lastly, we observe an intriguing behavior in the DINO methodologies. Despite fine-tuning on the galaxy dataset without labels, performance does not improve. By examining the attention maps of the CLS token in the last encoder block, we notice a significant difference between the non-fine-tuned and fine-tuned versions of DINO. However, this substantial variation in model attention does not translate into better classification performance. Several factors might contribute to this phenomenon:

1. Simplicity of k-Means Clustering: The k-means clustering algorithm used for classification might be too simplistic for capturing the complex and subtle differences between various galaxy morphologies. K-means clustering assumes spherical clusters of similar size, which might not align well with the intricate and nuanced structures of galaxies.
2. Subtlety of Morphological Differences: Differentiating between different types of galaxies often involves very subtle features, which might be challenging for a self-supervised training method that does not rely on labels. For instance, the distinction between spiral and edge-on galaxies is far more subtle than the difference between categories like cats and dogs.

These observations suggest that while DINO's self-supervised approach is powerful, it may require more sophisticated clustering techniques or additional domain-specific adjustments to effectively classify galaxy morphologies. Further research is needed to explore alternative unsupervised learning methods or enhancements to the current approach to better capture the nuanced differences in galaxy structures.

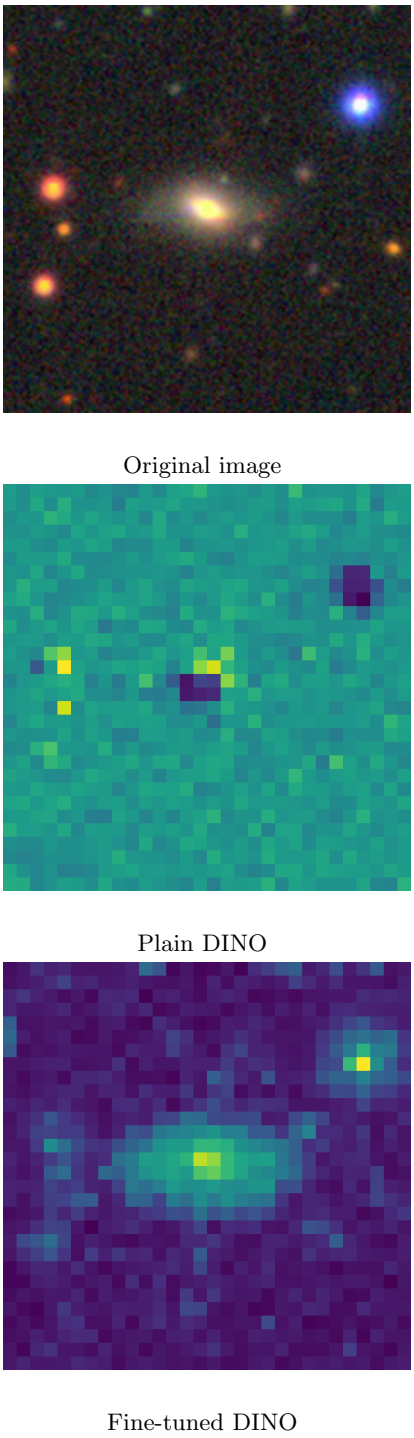

Figure 5: Example of attention maps (one head)

6. Limitations

This study was deliberately designed using simple and well-established models and techniques to ensure clarity in our comparative analysis. Our primary objective was not to develop the best pos-

sible model, but rather to enable the clearest and most understandable comparison of results.

To this end, we selected the VGG11 model due to its simplicity and robustness, making it an ideal candidate for our purposes. Furthermore, we chose not to employ any specific data augmentation techniques or advanced clustering methods. This approach allowed us to maintain the focus on the core performance metrics of the basic model without the confounding effects of additional complex preprocessing steps or algorithms.

By adhering to these methodological choices, we aimed to provide a transparent and straightforward evaluation that can be easily interpreted and replicated.

7. Conclusions

We conclude that both deep and shallow fine-tuning are effective approaches for model adaptation, each offering distinct advantages depending on the context. To determine the most appropriate method, developers should evaluate the degree of divergence between the fine-tuning dataset distribution and the pre-training distribution. If the distributions are significantly different, deep fine-tuning may be more beneficial, whereas shallow fine-tuning could suffice for more similar distributions.

For highly specific requirements, such as scenarios where inference speed is critical or where the application demands unique architectural considerations, creating a custom architecture from scratch becomes a compelling option. Custom architectures allow for optimization tailored to specific tasks, enhancing performance and efficiency in ways pre-trained models may not achieve.

However, it is not advisable to experiment with unsupervised classification using DINO embeddings for non-trivial tasks. Our findings indicate that these methods do not demonstrate improved performance in complex applications, likely due to their reliance on self-supervised learning which may not capture the intricate patterns necessary for such tasks.

Ultimately, the choice of fine-tuning strategy should be guided by the specific needs of the application, the characteristics of the datasets involved, and the performance requirements. By carefully considering these factors, developers can select the most suitable approach to achieve optimal results.

References

[1] R. Alberghi, *Dino with low rank adaption*, https://github.com/Rikyf3/dino_lora.

[2] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, *Emerging properties in self-supervised vision transformers*, 2021.

[3] henrysky, *Galaxy10 decals dataset*, https://github.com/henrysky/Galaxy10.

[4] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2015.

Appendix A. Additional Information

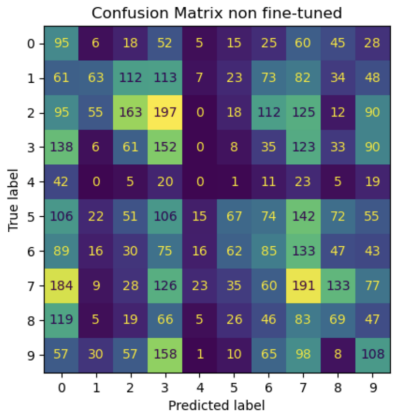

Figure A.1: Confusion matrix of untuned DINO with k-means head.

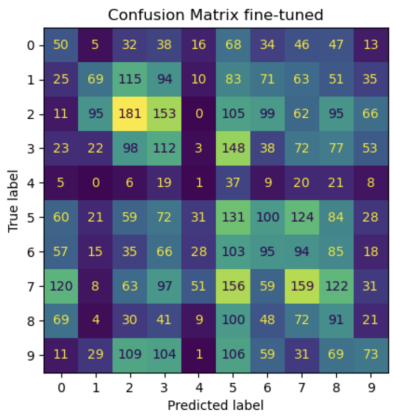

Figure A.2: Confusion matrix of fine-tuned DINO with k-means head.

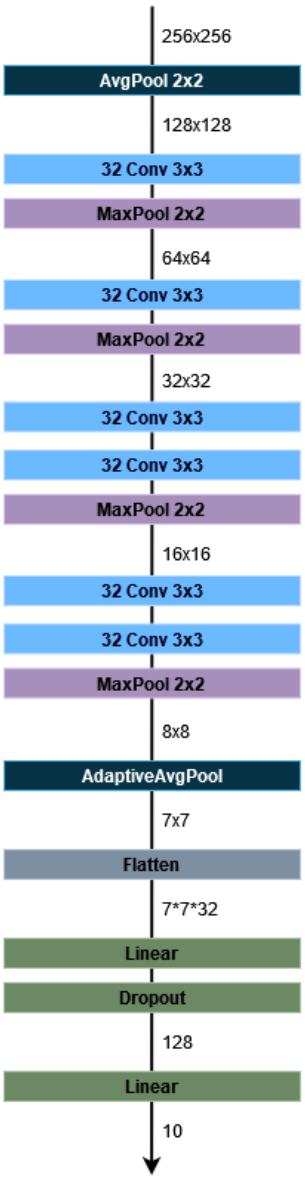

Figure A.3: Outline of our custom architecture. All convolutional and linear layers are followed by ReLU and BatchNorm.