UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

ERICK DA SILVA PULS

An evaluation of pre-trained models for feature extraction in image classification

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Engineering

Advisor: Prof. Dr. Joel Luís Carbonera

Porto Alegre
Abril 2023

“So long, and thanks for all the fish.”

— Douglas Adams

ACKNOWLEDGMENT

No one who achieves success does so without acknowledging the help of others. Therefore, first, I dedicate this monograph to my dear father, Claudio Valdir Puls (*in memoriam*), whose presence was essential in my life. And to my dear mother, Mara Lisiane da Silva Puls, whose commitment to educating me always came first - here are the results of her efforts.

With much gratitude and love, I also dedicate this work to my friends and family for all the support I have received. Thank you for the encouragement and for not letting me give up, even in the most difficult times. A special thanks to my sisters Audrey da Silva Puls and Camille Schneider Ribeiro, for always believing in me! The love you have for me is what encourages me to fight and win every day!

To Professor Joel Luís Carbonera, for accepting to supervise me even on short notice, and even so, guiding me throughout the development process of the present study. Moreover, to Matheus Vinícius Todescato for his patience in assisting my learning process regarding Python and the Geological Images dataset.

To my college friends, Leonardo Droves and William Roesch, who were great companions on the journey, and for the support they gave me throughout the course. To my friend, Guilherme Lamb, for several valuable programming suggestions. As they say - in the cookie of life, friends are the chocolate chips.

This research work was only possible through the support of my wife Grace dos Santos Feijó, who was able to inspire me every day, and whose presence has always positively affected my life in all aspects, and taught me every day how to be a better person. Thanks for helping me make this dream come true. This is one of the many achievements by your side. Love you. And, finally, to my dog and buddy Bruce, who is always by my side ready to cheer me up when I need it.

ABSTRACT

In recent years, we have witnessed a considerable increase in performance in image classification tasks. This performance improvement is mainly due to the use of *deep learning techniques*. Generally, deep learning techniques demand a large set of annotated data, making it a challenge to apply this method when little data is available. In this scenario, *transfer learning* strategies have become a promising alternative to overcome these issues. This work aims to compare the performance of different pre-trained neural networks for *feature extraction* in image classification tasks. We evaluated 16 different pre-trained models in four datasets, including a dataset of Geological Images that are the focus of this work. Our results demonstrate that for the Geological Images dataset, the best model was CLIP-ViT-B followed by CLIP-ResNet50. Similarly, the best general performance along all four datasets was achieved by CLIP-ViT-B and ViT-H-14, where the CLIP-ResNet50 model had similar performance but with lesser variability. Therefore, our study provides evidence supporting the choice of models for *transfer learning* in image classification tasks involving the four target datasets.

Keywords: Machine learning. Neural Networks. Image Classification. Transfer Learning. Feature Extraction. Geology.

ABSTRACT

Nos últimos anos, temos testemunhado um aumento considerável da performance em tarefas de classificação de imagens. Este aumento de performance se deve principalmente à utilização de técnicas de *deep learning*. Em geral, a aplicação de *deep learning* demanda um grande conjunto de dados anotados, o que torna desafiador aplicar tais técnicas em contextos com poucos dados anotados. Neste cenários, estratégias de *transfer learning* vêm se mostrando uma alternativa promissora para superar estes desafios. O objetivo deste trabalho é avaliar a performance de extração de *features* de diferetes redes neurais pré-treinadas aplicadas ao problema de classificação de imagens. Nossos resultados demonstraram queue para o dataset de Images Geológicas, o melhor modelo foi o CLIP-ViT-B, seguido do CLIP-ResNet50. Semelhante, a melhor performance geral dentre todos os datasets foi alcançada pelos modelos CLIP-ViT-B e ViT-H-14, onde o modelo CLIP-ResNet50 obteve perfomance semelhante, porém com variabilidade ainda menor. Sendo assim, nosso trabalho pode fornecer evidências que suportem a escolha de modelos para *transfer learning* em tarefas de classificação de imagens envolvendo o dataset de Imagens Geológicas.

Keywords: Machine learning. Neural Networks. Image Classification. Transfer Learning. Feature Extraction. Geology.

LIST OF ABBREVIATIONS AND ACRONYMS

ML	Machine Learning
AI	Artificial Intelligence
SML	Supervised Machine Learning
IC	Image Classification
ANN	Artificial Neural Network
DL	Deep Learning
CNN	Convolutional Neural Network
DNN	Deep Neural Network
ViT	Vision Transformer
FE	Feature Extraction
TL	Transfer Learning
CL	Convolutional Layers
ReLU	Rectified Linear Unit
CM	Confusion Matrix
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

The rapid advancements in technology in the last decades have pushed organizations to produce and accumulate all kinds of data. In the past, critical organizational information was primarily represented by structured data stored in databases. However, nowadays a significant part of this information is represented in an unstructured way, for instance as images (PFERD, 2010).

The industry of gas, energy, and biofuel is a billionaire sector that has an immense volume of information represented in unstructured ways. Therefore, there is a need to develop approaches capable of recovering and evaluating this type of information in this industry (PFERD, 2010). In that sense, one of the challenges concerning image recovery is that the semantic content of images is not apparent, so this information is not easily acquired through direct queries. An alternative for recovering images is annotating them first (HOLLINK et al., 2003) in a way that allows us to retrieve them by querying for the annotations. However, it is necessary to bear in mind that manual annotation of large databases of images is time-consuming and impractical. In this context, machine learning can be used to automatically classify these large databases of images, thus enabling retrieval through direct queries.

Machine learning (ML) has proved to be an important tool in the growth and expansion of several areas of knowledge, such as the biomedical (ALZUBAIDI et al., 2021), automotive (BORG et al., 2018), and industrial planning areas (CADAVID; PABLO et al., 2020). In Geosciences, ML has shown impressive results in the analysis of seismic attributes, facies classification, and other tasks (DOSOVITSKIY et al., 2020; MANIAR et al., 2018; KARPATNE et al., 2018). These recent developments encourage the application of state-of-the-art ML techniques for dealing with image classification and recovery in Geosciences.

Image Classification (IC) aims to classify the image as a whole by assigning a specific label to it. Usually, labels in an IC task refer to objects that appear in the image, kinds of images (photographs, drawings, etc.), feelings (sadness, happiness, etc.), etc (LANCHANTIN et al., 2021).

Most of the recent approaches for IC are based on deep neural network (DNN) architectures. This architecture usually demands a large set of annotated data, making it challenging to apply *deep learning* when small amounts of data are available. In this scenario, *transfer learning* strategies have become a promising alternative to overcome these

issues. One of the main alternatives of *transfer learning* is through *feature extraction*, where models that were trained on large datasets can be used for producing informative features that can be used by another classifier. By using *transfer learning*, we can leverage knowledge previously learned by neural network models on a large dataset and use this knowledge in a context where just small datasets are available.

There are currently several large datasets available, such as Imagenet (DENG et al., 2009), and a range of models that were pre-trained on these datasets [1]. The literature suggests that particular tasks on distinct datasets can benefit from different pre-trained models (MALLOUH; QAWAQNEH; BARKANA, 2019; ARSLAN et al., 2021). In this context, the main goal of this work is to compare and evaluate the performance of *feature extraction* (FE) of various pre-trained neural networks in the task of geological image classification. A secondary goal is to extend this comparison by including other well-known image datasets. This analysis can support the choice of models for FE in image classification tasks involving the Geological Images dataset and also, although in a more limited way, it can suggest reasonable model choices for other datasets.

In this study, Geological Images (TODESCATO et al., 2023), Stanford Cars (KRAUSE et al., 2013), CIFAR-10 (KRIZHEVSKY; HINTON et al., 2009), and STL10 (COATES; NG; LEE, 2011) datasets were adopted for analyzing the performance of FE of the following pre-trained models: AlexNet, ConvNeXt Large, DenseNet-161, GoogLeNet, Inception V3, MNASNet 1.3, MobileNet V3 Large, RegNetY-3.2GF, ResNeXt101-64x4D, ShuffleNet V2 X2.0, SqueezeNet 1.1, VGG19 BN, VisionTransformer-H/14, Wide ResNet-101-2, and both CLIP-ResNet50, and CLIP-ViT-B. In order to evaluate the performance of the considered pre-trained models, a set of metrics was selected: accuracy and macro and weighted averages of precision, recall, and F1-measure.

Our results indicate that the pre-trained models CLIP-ResNet50, CLIP-ViT-B, and VisionTransformer-H/14 had significantly better performance than the other pre-trained models for all datasets, including the dataset of Geological Images. It is important to notice that these are the only three models among those considered in our experiments that include transformers in their architecture, while the others are based solely on CNN architectures. Our analysis also indicates that there are differences regarding the pattern of performances of these three transformer-based architectures in comparison with the performances of the CNN-based architectures across the datasets, in all the considered metrics. These differences become evident when we analyze the Pearson correlation in

[1] Can be accessed through <https://pytorch.org/vision/stable/models.html>

4.2. Moreover, our analysis suggests that the Stanford Cars dataset is the most challenging of all datasets analyzed. We hypothesize that it is due to its large number of classes, few samples per class, and the inclusion of images with different sizes and features at different scales.

We discuss the theoretical background in Chapter 2 of this study. We also discuss some related works in Chapter 3. Then, we present our methodology and results in Chapter 4. Finally, our conclusions are presented in Chapter 5.

2 THEORETICAL BACKGROUND

The main concepts concerning this work will be presented in the following sections.

2.1 Machine Learning

Machine learning is a field of Artificial Intelligence (AI). The area of AI aims to enable machines to mimic intelligent behavior. In that sense, ML is a subfield of AI that studies methods in which computer systems learn and adapt upon experience (JOVEL; GREINER, 2021; RUSSELL, 2010; MITCHELL, 1997).

In the context of machine learning, algorithms build models from data during a training phase, and the resulting model is capable of making predictions on new observations. In this regard, the input data used during the training phase, in general, is a set of data samples, where each sample is characterized by a set of features.

There are three main subdivisions of machine learning (MITCHELL, 1997):

- Supervised learning (SML): In this approach, models are trained with labeled data sets, which are sets of samples associated with some class (in classification tasks) or continuous value (in regression tasks). The resulting model learns to map an input sample to a suitable output.
- Unsupervised learning: In this approach, the algorithm is applied to unlabeled datasets. The goal, in this sense, is finding patterns or trends, for example. This approach does not rely on labels for learning; it relies purely on the features of the data samples.
- Reinforcement learning: Different from the other two approaches, reinforcement learning does not need human-generated data for training. The algorithm learns through trial and error to take the best action according to a reward system.

In this study, we will focus on IC, which, in general, can be characterized as an SML task in the literature. Besides that, in this work, we are adopting artificial neural networks for performing IC.

2.2 Artificial Neural Networks

Artificial neural networks (ANNs), or neural networks, are a specific class of ML approaches. Originally, ANNs were formulated with a strong inspiration from the interconnected system of neurons in the human brain, in which thousands of processing nodes are interconnected and organized into layers (ALZUBAIDI et al., 2021; GOODFELLOW; BENGIO; COURVILLE, 2016; LECUN; HINTON, 2015).

The architecture of an ANN consists of layers of units (or neurons) connected to each other by weighted links (GOODFELLOW; BENGIO; COURVILLE, 2016). Each unit can process inputs and produce an output that is sent through the links to other units. Specifically, the input layer receives the input data and the output layer shows the result generated by the hidden layers. Figure 2.1 illustrates a simple neural network architecture built according to these principles.

Training ANN involves two main algorithms: back-propagation, and gradient descent. The back-propagation algorithm allows the information to flow backward after each forward pass through a network. Back-propagation refers to the method for computing the gradient, while the gradient descent is the process of descending through the gradient adjusting the parameters of the model to go down through the loss function (GOODFELLOW; BENGIO; COURVILLE, 2016). Gradient descent is an iterative algorithm, that starts from a given point on a function and travels down its slope in steps with the aim of reaching the lowest point of that function (GOODFELLOW; BENGIO; COURVILLE, 2016).

In a simplified way, DNNs are artificial neural networks with several layers of neurons and, in this context, deep learning (DL) is a set of machine learning approaches based on DNNs.

In conventional ML approaches, it is common to adopt pre-processing steps for feature engineering and feature selection on the raw input data, aiming to improve the learning process and, consequently, the model classification performance. On the other hand, some DL approaches offer advantages over conventional approaches, since they are able to directly learn a more convenient set of features to support the classification tasks (Figure 2.2). Thus, such techniques can be used directly on raw unstructured data, eliminating the need for feature engineering and feature selection. These advantages are widely explored in the context of IC (ALZUBAIDI et al., 2021; GOODFELLOW; BENGIO; COURVILLE, 2016).

Figure 2.1: Visual representation of a simple Artificial Neural Network architecture. The units are represented by individual circles and are divided by color in each layer. Blue circles for the input layer, grey circles for the hidden layer, and red circles for the output layer. Each circle is linked to each other through colorful lines representing the different weighted links between the layers

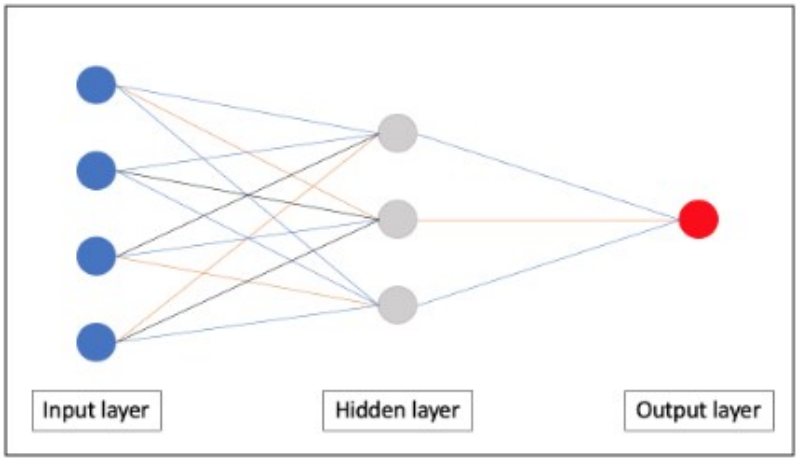
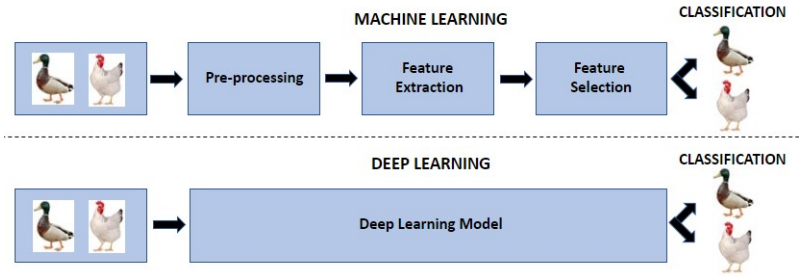


Figure 2.2: Illustration of the difference between deep learning and traditional machine learning



While ANNs might have a few hidden layers, a DL network can have hundreds of layers. The increasing of the number of different layers and nodes may increase the accuracy of a network, however, in general, this also implies that DL requires more computing power and data during the learning process (YAMASHITA et al., 2018; LECUN; HINTON, 2015).

There are several different DNN architectures proposed in the scientific literature. In the last decade, convolutional neural networks (CNNs) and vision transformers (ViTs) have shown promising results in IC tasks.

CNNs are a class of ANN architectures. They were inspired by the organization of the visual cortex of the human brain, where different individual neurons of the visual cortex are responsible for recognizing a small portion of the visual field, and the overlapping of those images that creates complex images (JOVEL; GREINER, 2021). Thus, CNN is a highly hierarchical approach, with specialized neurons and layers that are capable of detecting complex patterns, such as faces, for example.

The main characteristic of CNNs is the convolutional layers (CL), which specify a trainable set of filters that perform convolutional operations on image areas. These operations generate more suitable features for the IC task. These layers allow CNN to adaptively learn through backpropagation how to extract informative spatial features from an image. This allows us, for example, to identify objects accurately, the location of objects, as well as their relationship with other objects in an image (YAMASHITA et al., 2018).

ViT is a transformer architecture applied to computer vision that adopts the notion of attention (the relationship between pairs of input tokens) to process information across images. When used directly to sequences of image sets, it shows an impressive performance in IC cases (DOSOVITSKIY et al., 2020). Also when pre-trained on large quantities of data, ViT reaches excellent results even when compared to state-of-the-art convolutional networks (DOSOVITSKIY et al., 2020).

Since training DNNs from scratch requires large labeled datasets, it can be a challenge in certain domains in which data acquisition is hard. In this sense, transfer learning emerges as a promising approach for dealing with this scenario (ZHUANG et al., 2020).

2.3 Transfer Learning

Transfer learning (TL) is an ML approach inspired by the human ability to transfer knowledge across tasks. Thus, TL intuitively involves transferring the comprehension of a previously learned related task to a new one to improve learning in the new task (ZHUANG et al., 2020; SHRESTHA; MAHMOOD, 2019). This strategy has been applied in a variety of tasks, including IC (YAMASHITA et al., 2018).

As previously stated, DL networks are trained using a huge amount of data to learn suitable weights and biases during the training process. After training, the patterns learned by the network represented in its weights can be *transferred* in different ways for different tasks (ALZUBAIDI et al., 2021). Therefore, the advantage of transfer learning lies in its power to pre-train a model on one sort of information (for which there is a sufficient amount of labeled data available) and apply the learned patterns to different tasks for which only small datasets are available.

The two main approaches of TL are *fine-tuning* and *feature extraction* (LI; HOIEM, 2017). The fine-tuning approach involves training an initial model with large volumes of data and, in the next step, refining the model by training it on a smaller specific dataset

(ZHUANG et al., 2020; SHRESTHA; MAHMOOD, 2019) reusing the weights and biases of the pre-trained model as the initial condition for training it for the new task. On the other hand, FE, which will be the focus of this work, only reuses the weights and biases of the pre-trained architecture, without refining them in a subsequent learning process. Thus, parts of the pre-trained architecture are used only for extracting relevant features from the input data. In this context, these relevant parts of pre-trained models play the role of *feature extractors* that can be integrated into a novel architecture. In this sense, the resulting architecture can be trained in a smaller dataset, but keeping the weights and biases of the feature extractor in their original form, without updating them during the training. For applying FE, a common practice involves replacing the last layer of the pre-trained model (responsible for performing the classification) with a novel layer with a suitable number of output units for the task and trainable parameters that are updated during the training process (LI; HOIEM, 2017).

Thus, the main difference between these two TL approaches is that in fine-tuning the entire network is kept free (trainable) and retrained in the new task, starting from the weights it had in the original task. When doing FE, the pre-trained part of the network responsible for extracting features is not re-trained; it is maintained "frozen" during the training of the new network, performing only FE (LI; HOIEM, 2017; DARA; TUMMA, 2018).

2.4 Pre-trained Models

There is a range of pre-trained models available in repositories[1]. Bellow, we will provide a brief description of the pre-trained models considered in this work. Their description will be summarized since it is not included in the scope of this work to discuss the details of each model.

The majority of the models considered in this work were pre-trained using ImageNet-1K[2](DENG et al., 2009) dataset and two of these models were pre-trained using a dataset called YFCC100M[3](THOMEE et al., 2016). Table 2.1 presents the following properties of the selected models: number of output features, number of parameters, training dataset, and architecture.

[1]Can be accessed through <https://pytorch.org/vision/stable/models.html>
[2]Can be accessed through <https://image-net.org/index.php>
[3]Can be accessed through <http://projects.dfki.uni-kl.de/yfcc100m/>

Table 2.1: Pre-trained Models Information

Models	Output features	Parameters	Training dataset	Architecture
alexnet	256	61,100,840	ImageNet-1K	CNN
clip_rn50	1024	63,000,000	YFCC100M	CNN + Transformer
clip_vit_b	512	63,000,000	YFCC100M	Transformer + Transformer
convnext_large	1536	197,767,336	ImageNet-1K	CNN
densenet161	2208	28,681,000	ImageNet-1K	CNN
googlenet	1000	6,624,904	ImageNet-1K	CNN
inception_v3	1000	27,161,264	ImageNet-1K	CNN
mnasnet1_3	1000	6,282,256	ImageNet-1K	CNN
mobilenet_v3_large	960	5,483,032	ImageNet-1K	CNN
regnet_y_3_2gf	1000	19,436,338	ImageNet-1K	CNN
resnext101_64x4d	1000	83,455,272	ImageNet-1K	CNN
shufflenet_v2_x2_0	1000	7,393,996	ImageNet-1K	CNN
squeezenet1_1	512	1,235,496	ImageNet-1K	CNN
vgg19_bn	512	143,678,248	ImageNet-1K	CNN
vit_h_14	1000	632,045,800	ImageNet-1K	Transformer
wide_resnet101_2	1000	126,886,696	ImageNet-1K	CNN

AlexNet (eg., alexnet) [4]: It is a deep convolutional neural network architecture with more than 60 million parameters, consisting of five CL, some of which are followed by max-pooling layers, and three fully connected layers with a final 1000-way softmax (KRIZHEVSKY, 2014). AlexNet also introduced the concept of using rectified linear unit (ReLU) activation functions instead of traditional sigmoid activation functions (eg., alexnet)

CLIP-ResNet50 (eg., clip$_rn50$**)** [5]: Its base model uses a ResNet50 modified version as its image encoder and a masked self-attention Transformer as its text encoder. These encoders are trained to maximize the similarity of pairs of image and text using a contrastive loss technique. The model uses ResNet50 (HE et al., 2016) as the base architecture for the image encoder and replaces the global average pooling layer with an attention pooling mechanism. This mechanism is implemented as a single layer of transformer-style multi-head attention where the query is conditioned on the global average-pooled representation of the image (RADFORD et al., 2021).

CLIP-ViT-B (eg., clip$_vit_b$ [6]: It is similar to the CLIP-ResNet50 architecture, where the ResNet image encoder is replaced for a Vision Transformer with a B4 backbone. The B4 backbone refers to a specific variation of the ViT architecture with larger image resolution and more parameters compared to the original model, making it more powerful than the original ViT model. The ViT has an additional layer normalization to the combined patch and position embeddings before the transformer

[4]Can be accessed through <https://pytorch.org/vision/stable/models/alexnet.html>
[5]Can be accessed through <https://github.com/openai/CLIP/blob/main/model-card.md>
[6]Can be accessed through <https://github.com/openai/CLIP/blob/main/model-card.md>

and uses a slightly different initialization scheme (DOSOVITSKIY et al., 2020). The text encoder is also a transformer (VASWANI et al., 2017). As a base size, they use a 63M-parameter, 12-layer, 512-wide model with 8 attention heads (RADFORD et al., 2021).

ConvNeXt Large (eg., convnext$_{large}$) [7]: A larger and more powerful version of the ConvNext DL architecture (LIU et al., 2022) that achieves high accuracy in image recognition tasks by increasing the number of CL and parameters in the model (LIU et al., 2022). This model consists of multiple blocks of CL that extract increasingly complex features from an input image. Each block is composed of multiple CL with batch normalization and non-linear activation functions in between. The architecture also uses a technique called skip connections, where inputs are added to outputs from earlier layers in the model. This allows the model to maintain information from earlier stages of processing and can help prevent the vanishing gradient problem that can occur in DNNs.

DenseNet-161 (eg., densenet161) [8]: This model has a dense connectivity pattern, where each layer in the model is directly connected to every other layer in a feed-forward fashion (HUANG et al., 2017). This allows the model to learn more complex features by reusing and combining the outputs from earlier layers in the network. The architecture consists of multiple dense convolutional blocks, where each block contains multiple CL with batch normalization and non-linear activation functions in between. The output of each block is fed into a transition layer, which reduces the spatial dimension of the feature maps before passing them to the next dense block. DenseNet-161 is particularly effective at recognizing fine-grained details in images, making it useful for applications such as medical imaging, natural scene classification, and object detection (HUANG et al., 2017).

GoogLeNet (eg., googlenet) [9]: The key innovation in GoogleNet is its Inception module, which consists of multiple CL with different filter sizes that are combined in parallel (SZEGEDY et al., 2015). This allows the model to capture both fine-grained and high-level features in an image at the same time, improving its accuracy. GoogLeNet also uses a technique called global average pooling, where the output of the last convolutional layer is pooled to a single value for each feature map. This reduces the number of parameters in the model and helps prevent over-

[7] Can be accessed through <https://pytorch.org/vision/stable/models/convnext.html>
[8] Can be accessed through <https://pytorch.org/vision/stable/models/densenet.html>
[9] Can be accessed through <https://pytorch.org/vision/stable/models/googlenet.html>

fitting. The architecture consists of multiple Inception modules, with each module followed by a pooling layer and a batch normalization layer. The output of the final pooling layer is fed into a fully connected layer, which produces the classification output.

Inception V3 (eg., inception$_v3$) [10]: The key innovation in Inception V3 architecture is the adoption of factorization techniques to reduce the number of parameters in the model (SZEGEDY et al., 2016). This allows the model to be more efficient and faster to train. In addition, Inception V3 uses batch normalization, which helps the model to converge faster during training and prevents overfitting. The architecture consists of multiple Inception modules, where each module consists of multiple branches with different filter sizes and pooling operations. The outputs of these branches are then concatenated and fed into the next layer. Inception V3 also uses a technique called auxiliary classifiers, where smaller classifiers are inserted at intermediate layers of the network. These classifiers provide additional gradients during training and help to regularize the model.

MNASNet 1.3 (eg., mnasnet1$_3$) [11]: MNASNet 1.3 was designed as a DL architecture for mobile devices (TAN et al., 2019). It was developed as a result of a search algorithm that automatically discovers the optimal architecture, given a set of constraints such as model size and latency. The final architecture consists of multiple blocks, where each block contains multiple depthwise-separable CL with batch normalization and ReLU activation functions in between (TAN et al., 2019). This allows the model to learn complex features while minimizing the number of parameters and computations required. MNASNet 1.3 also uses a technique called Squeeze and Excitation blocks, which adaptively recalibrates the feature maps based on their importance, further improving the accuracy of the model.

MobileNet V3 Large (eg., mobilenet$_v3_large$) [12]: MobileNet V3 Large is a DL architecture designed for mobile devices and optimized for efficient computation and a small memory footprint (HOWARD et al., 2019). The key innovation is the use of a combination of depthwise and pointwise convolutions, which reduces the computation required while maintaining accuracy. The architecture of MobileNet V3 Large was developed by applying an architecture search algorithm that discovers the optimal architecture for the model given a set of constraints such as model size and la-

[10] Can be accessed through <https://pytorch.org/vision/stable/models/inception.html>
[11] Can be accessed through <https://pytorch.org/vision/stable/models/mnasnet.html>
[12] Can be accessed through <https://pytorch.org/vision/stable/models/mobilenetv3.html>

tency (HOWARD et al., 2019). The architecture consists of multiple blocks, where each block contains a combination of depthwise and pointwise convolutions with activation functions and batch normalization in between. The model also includes a feature fusion module that combines features from different layers to improve accuracy (HOWARD et al., 2019). MobileNet V3 Large also uses a technique called the h-swish activation function, which is a faster alternative to the widely used ReLU activation function. This further reduces computation requirements and improves performance.

RegNetY-3.2GF (eg., regnet$_{y32gf}$**)** [13]: RegNetY stands for "Regularized Network" and 3.2GF refers to the number of floating-point operations per second (FLOPS) the model requires for inference (RADOSAVOVIC et al., 2020). The resulting architecture is built by a regularized design space search algorithm that finds an optimal architecture. The architecture consists of multiple stages, where each stage contains multiple blocks with different numbers of channels and different levels of regularized scaling factors (RADOSAVOVIC et al., 2020). The regularized scaling factors ensure that the model is efficient and scalable, and the number of channels determines the depth and width of the model.

ResNeXt101-64x4D (eg., resnext101$_{64x4d}$**)** [14]: ResNeXt101-64x4D is a variant of the popular ResNet architecture. Its key innovation is the use of a group convolutional approach to increase the model's capacity and performance (XIE et al., 2017). This approach divides the input channels into groups and applies a separate convolutional operation to each group. This allows the model to learn more diverse and complex features while using fewer parameters than traditional CL. The architecture consists of multiple blocks, where each block contains multiple CL with batch normalization and ReLU activation functions in between (XIE et al., 2017). The number of groups and the number of channels in each block can be adjusted to optimize the model's performance and efficiency.

ShuffleNet V2 X2.0 (eg., shufflenet$_{v2_x2_0}$**)** [15]: ShuffleNet V2 X2.0 is a variant of the ShuffleNet architecture. Its key innovation is the use of a channel shuffle operation that mixes feature maps from different channels to reduce computational complexity and improve performance (MA et al., 2018). The architecture also uses depthwise convolution, which applies a single filter to each input channel instead of using

[13] Can be accessed through <https://pytorch.org/vision/stable/models/regnet.html>
[14] Can be accessed through <https://pytorch.org/vision/stable/models/resnext.html>
[15] Can be accessed through <https://pytorch.org/vision/stable/models/shufflenetv2.html>

multiple filters, further reducing computation requirements. The architecture consists of multiple stages, where each stage contains multiple ShuffleNet units (MA et al., 2018). Each ShuffleNet unit consists of a pointwise convolution, depthwise convolution, channel shuffle operation, and another pointwise convolution.

SqueezeNet 1.1 (eg., squeezenet1$_1$) [16]: SqueezeNet 1.1 is designed to have a small memory footprint and low computational complexity. Its key innovation is the use of fire modules, which are composed of squeeze and expand layers (IANDOLA et al., 2016). The squeeze layer uses 1×1 convolutions to reduce the number of input channels, while the expand layer uses both 1×1 and 3×3 convolutions to increase the number of output channels. This approach reduces the number of parameters required while maintaining accuracy. SqueezeNet 1.1 also uses techniques such as network pruning and regularization to further reduce the model's memory footprint and computational complexity. This allows it to achieve comparable accuracy to larger models while requiring significantly fewer parameters and computation requirements. The architecture consists of multiple fire modules, interleaved with max pooling and dropout layers (IANDOLA et al., 2016). The final layer is a global average pooling layer followed by a softmax classifier.

VGG19 BN (eg., vgg19$_bn$) [17]: VGG19 BN is an extension of the VGG16 architecture. The "BN" in VGG19 stands for "Batch Normalization", a technique that normalizes the inputs to each layer of the network to improve training stability and performance (SIMONYAN; ZISSERMAN, 2014). This technique is applied after every convolutional layer in the VGG19 BN architecture. Its architecture consists of 19 layers, including 16 CL and 3 fully connected layers. The CL use 3x3 filters with a stride of 1 and padding of 1, while the fully connected layers have 4096 units each. The architecture also includes max pooling layers after some of the CL (SIMONYAN; ZISSERMAN, 2014). One of the key features is its use of a uniform architecture throughout the network, with all CL having the same number of filters and the same filter size. This makes it easy to train and optimize the network.

Vision Transformer-H/14 (eg., vit$_{h1}4$) [18]: Vision Transformer-H/14 is an extension of the original ViT architecture. The "H" in ViT-H/14 stands for "huge", indicating that this model is larger and more powerful than the original ViT. The "14" refers to the number of layers in the network. ViT-H/14 is designed to handle high-resolution

[16] Can be accessed through <https://pytorch.org/vision/stable/models/squeezenet.html>
[17] Can be accessed through <https://pytorch.org/vision/stable/models/vgg.html>
[18] Can be accessed through <https://pytorch.org/vision/stable/models/vision_transformer.html>

images and has a total of 634 million parameters, making it one of the largest available models for dealing with images (DOSOVITSKIY et al., 2020). Like other transformer-based models, ViT-H/14 uses a self-attention mechanism to process images. It first splits the input image into a set of patches, which are then fed into a sequence of transformer blocks. Each block consists of a multi-head attention mechanism and a feed-forward neural network. The multi-head attention mechanism allows the model to attend to different parts of the input sequence and capture long-range dependencies (DOSOVITSKIY et al., 2020). In addition, ViT-H/14 uses a hybrid approach that combines both convolutional and transformer-based architectures. It applies a small number of CL to the input image before feeding it into the transformer blocks. This helps the model capture low-level features and reduce computational costs.

Wide ResNet 101-2 (eg., wide$_resnet101_2$**)** [19]: Wide ResNet 101-2 is a variant of the ResNet architecture (ZAGORUYKO; KOMODAKIS, 2016). The "101" refers to the number of layers in the network. At the same time, the "2" indicates that the network is wider than the original ResNet architecture, with a wider block structure that includes two CL in each residual block. The network architecture consists of a series of residual blocks, which allow the network to learn deep representations of the input image. Each residual block includes two CL followed by batch normalization and ReLU activation function. The output of each block is added to the input of the block, creating a shortcut connection that helps to propagate gradients and improve training. It also incorporates several other features to improve its performance, including dropout regularization, pre-activation, and strided convolution.

2.5 Model Evaluation

This section discusses metrics and techniques that can be applied to evaluate ML classifiers and that will be adopted in this work.

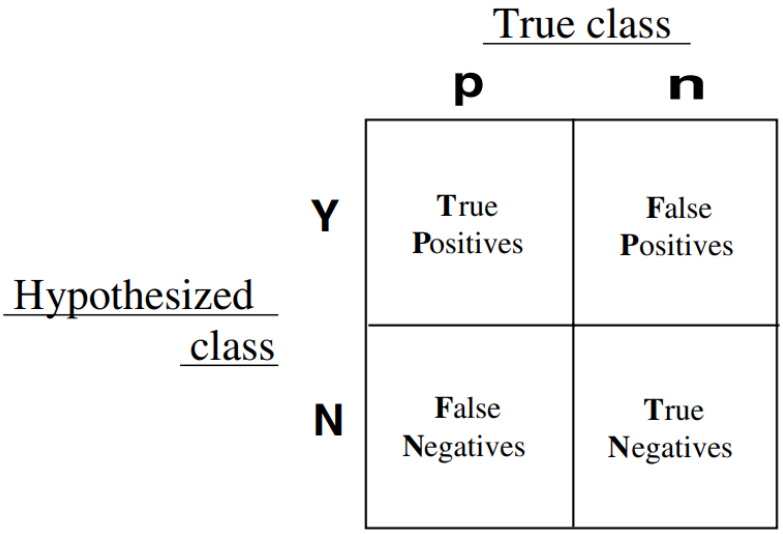
[19] Can be accessed through <https://pytorch.org/vision/stable/models/wide_resnet.html>

2.5.1 Performance Metrics

Here we present the metrics that will be adopted to evaluate the performance of our models: accuracy, precision, recall, and F1-score. Since we are dealing with multiclass classification problems, in this context, precision, recall, and F1-score will be considered with a macro and weighted average (VANI; RAO, 2019).

The adopted metrics are calculated according to a confusion matrix (CM), which shows the relationship between *actual* and *predicted* values in a classification problem. The evaluation metrics are defined by the following prediction categories: *True positives* (TP), which correctly indicates the presence of a condition or characteristic; *True negatives* (TN), which correctly shows the absence of a condition or characteristic; *False positive* (FP), which wrongly indicates that a particular condition or attribute is present; *False negative* (FN), which wrongly indicates that a particular condition or attribute is absent. A schema of a confusion matrix for a binary classification problem is represented in Figure 2.3. For multiclass classification problems with N classes, the resulting confusion matrix is a $N \times N$ matrix, whose lines and columns represent each class, and the prediction categories previously discussed are calculated class-wisely.

Figure 2.3: Schema of a confusion matrix for a binary classification problem. Where **Y** stands for yes, **N** for no, **p** for positive, and **n** for negative



2.5.2 Accuracy

Accuracy computes the proportion of correctly classified observations out of all observations.

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

2.5.3 Precision

Precision is calculated as the ratio between true positives and the sum of true positives and false positives. A model that has no false positives shows a precision of 1.0.

$$Precision = \frac{TP}{TP + FP}$$

2.5.4 Recall

Recall is the ratio of correctly predicted positive observations to all the observations originally classified as yes. A model that shows no false negatives has a recall of 1.0.

$$Recall = \frac{TP}{TP + FN}$$

2.5.5 F1-score

F1-score is a measure that deals with both false positives and false negatives. It combines the precision and recall of a classifier into a single metric by taking their harmonic mean.

$$F1 - score = \frac{2 \times (Recall \times Precision)}{(Recall + Precision)}$$

2.5.6 Macro and Weighted Averages

In this work, we adopt two main methods of calculating precision, recall, and F1-score for multi-class classification tasks. These two methods involve different ways of aggregating results on the CM in multiclass classification problems.

2.5.6.1 Macro Average

This method treats all classes equally regardless of their support values. The macro average is calculated individually for each performance measure as follows.

The macro-average of Precision is calculated by taking the average of all precision values calculated for each class:

$$Precision_{Macro} = \frac{\Sigma_{i=1}^{n} P_i}{n},$$

where n is the count of classes in the dataset and $P_i$is the precision value of the i-th class.

The macro-average of Recall is calculated by taking the average of all Recall values calculated for each class:

$$Recall_{Macro} = \frac{\Sigma_{i=1}^{n} R_i}{n},$$

where n is the number of classes in the dataset and R_i, is the recall value calculated for the i-th class.

The macro-average of the F1-score is the harmonic median of the Macro-average of Recall and Precision.

$$F1 - score_{Macro} = \frac{\Sigma_{i=1}^{n} F_i}{n},$$

where n is the number of classes in the dataset and F_i, is the recall value calculated for the i-th class.

2.5.6.2 Weighted-average

The weighted average of the F1-score takes the mean of all per-class F1-score weighted by each class's support.

$$F1 - score_{Weighted} = \frac{\Sigma_{i=1}^{n} F_i \times C_i}{C},$$

where F_i, is the F1-score calculated for the i-th class, C_i is the number of instances of the i-th class, C is the total number of instances and n is the number of classes in the dataset.

Similarly, we can calculate Weighted-precision and Weighted-recall:

$$Precision_{Weighted} = \frac{\Sigma_{i=1}^{n} P_i \times C_i}{C},$$

where P_i is the Precision calculated for the i-th class, C_i is the number of instances for i-th class.

$$Recall_{Weighted} = \frac{\Sigma_{i=1}^{n} R_i \times C_i}{C}$$

, where R_i is the Recall calculated for the i-th class.

2.5.7 Cross Validation

Cross-validation is a process that can be used to estimate the quality of a classification model (such as a trained neural network). It also allows for comparing and selecting an appropriate model for the specific problem. When applied to different classifiers with different free parameter values (such as the number of hidden nodes, back-propagation learning rate, etc), the results of cross-validation can be used to select the best set of parameter values. These procedures are based on the idea of repeating the training and testing on different randomly chosen subsets or splits of the original dataset.

The most common cross-validation is the *k-fold cross-validation* procedure. In this process, the dataset is split into k-folds, and a process of training classifiers and evaluating test data is performed k times. At each iteration of the process, some fold is used as a test set and the remaining $k-1$ folds are used as the training set. In this process, every sample in the data will be included in the test set at some step. The performance of the model may then be estimated by taking the average performance across the k-folds used as a test set (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.6 Datasets

Image datasets are collections of digital images that can be both annotated or not. In general, image datasets can be used for various computer vision tasks, such as image recognition, object detection, segmentation, object tracking, etc. These datasets may contain images of different sizes, resolutions, and formats, and they are often annotated with additional information such as object classes, bounding boxes, segmentation masks, etc. They are essential for training and evaluating computer vision models, and their availability has enabled significant advances in the field of computer vision in recent years. These datasets are often used to benchmark new algorithms and to improve the performance of existing ones.

There are several widely used image datasets in computer vision research. In this work, the following ones were considered for image classification, since they are widely used in the literature, are colorful, and have different characteristics: Geological Images dataset (TODESCATO et al., 2023), Stanford Cars (KRAUSE et al., 2013), CIFAR-10 (KRIZHEVSKY; HINTON et al., 2009), and STL10 (COATES; NG; LEE, 2011).

The main focus of this work is to evaluate the performance of the pre-trained models over the Geological Images dataset. However, as a secondary objective, Stanford Cars, CIFAR-10, and STL10 datasets were also included in our evaluation, in order to carry out a more comprehensive evaluation of the models. Table 2.2 shows the main information of all datasets used in this work.

Table 2.2: Datasets Information

Dataset	Instances	Classes	Average Instances per Class $\pm$ Std
Geological Images (TODESCATO et al., 2023)	25725	45	571,67 $\pm$ 1290,90
Stanford Cars (KRAUSE et al., 2013)	16185	196	84 $\pm$ 6,28
CIFAR-10 (KRIZHEVSKY; HINTON et al., 2009)	60000	10	6000 $\pm$ 0
STL10 (COATES; NG; LEE, 2011)	100000	10	10000 $\pm$ 0

Regarding the selected datasets, CIFAR-10 and STL10 are balanced and include sets of images of homogeneous size. The Geological images and the Stanford Cars datasets are unbalanced (Stanford Cars is slightly unbalanced) and have images of heterogeneous sizes. In the following, we will provide more details regarding each dataset adopted in this study.

2.6.1 Geological Images dataset

This is a domain-specific dataset (ABEL et al., 2019) that includes a set of annotated images that are relevant for applications in Geosciences. The adopted image dataset includes images extracted from public and private sources. The images were obtained from the Petrobras Geoscience Bulletins, Theses, and Dissertations from the Geosciences Institute of UFRGS, as well as from several sources on the internet. Cleaning procedures were performed to refine the set of images, aiming to remove invalid and/or non-relevant images. In total, the dataset includes 25,725 annotated images of 45 different classes. This dataset presents an imbalance in the distribution of images across classes, where there are classes with only 36 images and classes with 8450 images. The average of images per class is 571,67 with a standard deviation of 1290,90. The distribution of instances per class of this dataset is represented in Figure 2.4. Figure 2.5 shows a set of images that are included in the Geological Images dataset. Besides that, the images included in this dataset are heterogeneous regarding height, width, and aspect ratio, as is represented in Figure 2.6.

Figure 2.4: Illustration of the imbalance in the number of images for each class on the Geological Images dataset (TODESCATO et al., 2023).

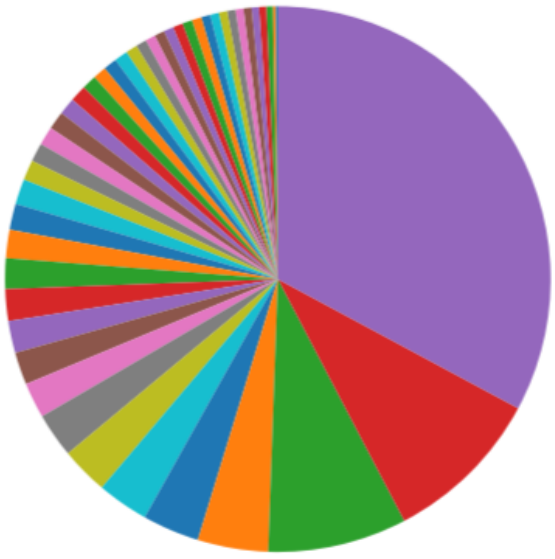


Figure 2.5: Examples of images included in the Geological Images dataset (TODESCATO et al., 2023).

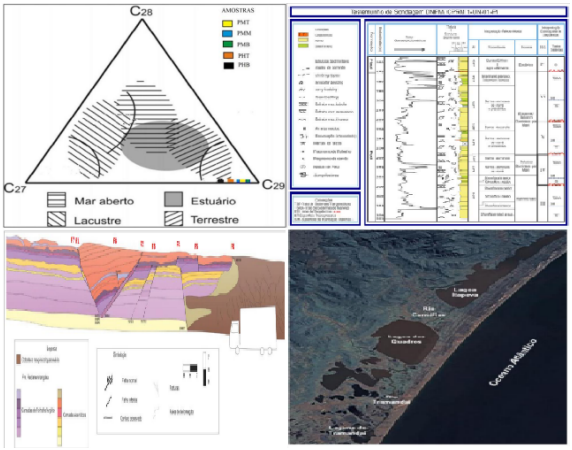
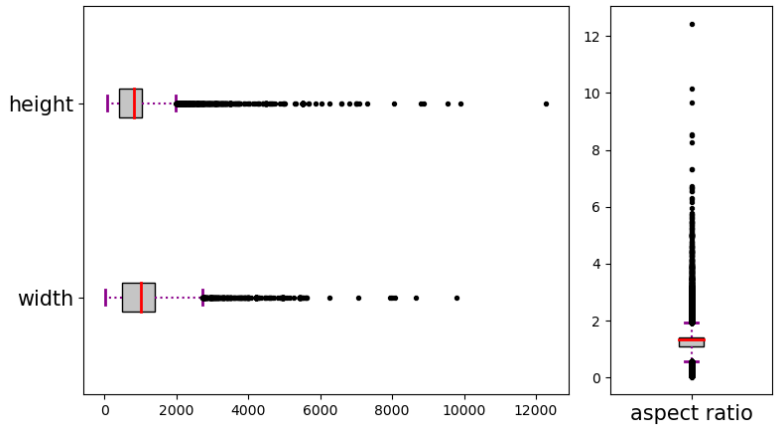


Figure 2.6: Statistics of the distribution of height, width, and aspect ratio of images in the Geological Images dataset (TODESCATO et al., 2023).



2.6.2 Stanford Cars

It consists of 16,185 images of 196 different car classes, where each car class has been carefully labeled with its make, model, and year. The images were collected from various online sources, including auction websites, car dealerships, and online classifieds. Each image in the dataset [20] is associated with a text file that contains metadata about the car, including the car's make, model, and year, as well as its class ID, which ranges from 1 to 196. The dataset also includes a set of bounding boxes for each image, which identify the location of the car within the image. Figure 2.8 shows a set of images that are included in the Stanford Cars dataset. Also, the images included in Stanford Cars dataset compared to Geological Images dataset are partially heterogeneous regarding height, width, and aspect ratio, as is represented in Figure 2.7.

2.6.3 CIFAR-10

The dataset [21] contains 60,000 32x32 color images in 10 different classes. Each class contains 6,000 images, and the dataset is split into 50,000 training images and 10,000 test images. The classes in the dataset are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Figure 2.9 shows a set of images that are included in the CIFAR-10 dataset.

[20] <https://ai.stanford.edu/~jkrause/cars/car_dataset.html>
[21] <https://www.cs.toronto.edu/~kriz/cifar.html>

Figure 2.7: Statistics of the distribution of height, width, and aspect ratio of images in the Stanford Cars dataset (TODESCATO et al., 2023).

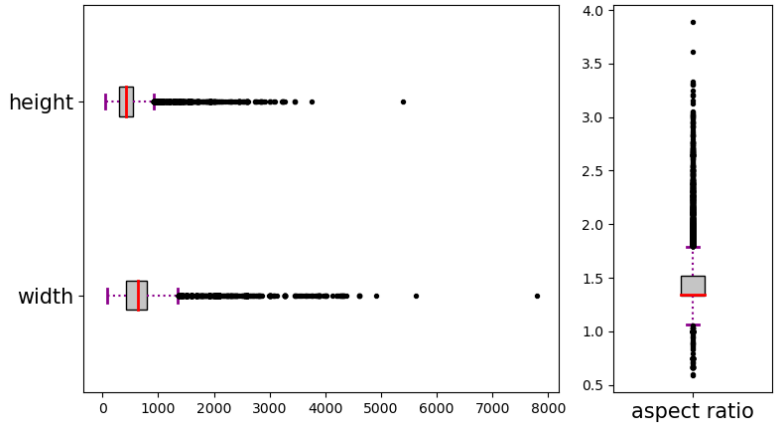
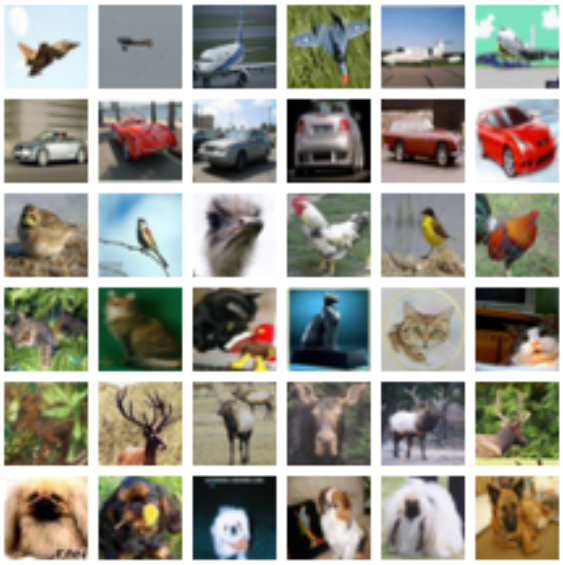


Figure 2.8: Images included in the Stanford Cars dataset (KRAUSE et al., 2013).

Figure 2.9: Images included in the CIFAR-10 dataset (KRIZHEVSKY; HINTON et al., 2009).



2.6.4 STL10

This is an image recognition dataset[22] that contains a set of 10 classes, each represented by 1,000 labeled images. The images in the dataset are 96x96 pixels and are derived from a larger set of unlabeled images collected from the Internet. STL10 was developed to be a more challenging dataset for image recognition tasks, as the images are larger than those in other popular datasets like CIFAR-10 and ImageNet. The 10 classes are airplane, bird, car, cat, deer, dog, horse, monkey, ship, and truck. It is also divided into a training set of 5,000 images and a test set of 8,000 images, with each class evenly represented in both sets. In addition to the labeled images, the STL10 dataset also includes an unlabeled set of 100,000 images, which can be used for unsupervised learning tasks like FE and clustering. In this work, we adopted only the labeled images. Figure 2.10 shows a set of images that are included in the STL10 dataset.

Figure 2.10: Images included in the STL10 dataset (COATES; NG; LEE, 2011).

[22] <https://cs.stanford.edu/~acoates/stl10/>

3 RELATED WORKS

The TL approach based on FE has been adopted for IC in several domains, such as Biomedicine (ALZUBAIDI et al., 2021) and Geology (DOSOVITSKIY et al., 2020; MANIAR et al., 2018; KARPATNE et al., 2018). In this work, we reviewed the literature covering the last five years and focused on comparing the performance of FE for different pre-trained models. In total, we selected eighteen papers for our literature overview, of which five are specific to IC in Geology. In our literature review, twenty-three different models were found, among which the most recurrent pre-trained models were the VGG16 and the Inception V3, both for studies in general and those specific to Geology, and the third most used model was the AlexNet. Of the eighteen articles, only two did not use the ImageNet dataset for pre-training the models. Therefore, we can observe that ImageNet is one of the most used datasets for pre-training models for IC.

There is a range of pre-trained models that can be applied for transfer learning. The main expected result of FE from the pre-trained models is to improve classification quality. The size and similarity of the target dataset and the source task can be used to choose the pre-trained model (FAWAZ et al., 2018).

The literature suggests that each dataset may need a different pre-trained model. For instance, for plankton classification (LUMINI; NANNI, 2019), when adopted as a feature extractor the Inception V3, AlexNet, VGG16, VGG19, ResNet50, ResNet101, DenseNet-161, and GoogLeNet pre-trained models with three different datasets, the DenseNet-161 model presented the best results. On the other hand, when classifying pathological brain images, Kaur & Gandhi (2020) found that among eight pre-trained models, the AlexNet showed the best results.

Finally, using the CIFAR-10 dataset, and experimenting with the Inception V3, GoogLeNet, SqueezeNet 1.1, and DarkNet53, ShuffleNet models, Kumar, Anuar and Hassan (2022) found that, overall, the Inception V3 model achieved the highest accuracy, as well as higher values in other evaluation metrics including precision, sensitivity, specificity, and F1-score (KUMAR; ANUAR; HASSAN, 2022).

In summary, for specific application needs, finding the right pre-trained model can be challenging. Different models can present better results for different datasets and different parameters of performance (BAKER; ZENGELER; HANDMANN, 2022). Therefore, it is essential to systematically investigate the usability of several pre-trained models to find the best match for specific datasets.

Although ML has been successfully applied in Geosciences in the last years, TL for IC in this domain is still not as much explored as in other areas (LIMA et al., 2019).

Recent papers are showing the capacity of DL and TL to facilitate the analysis of uninterpreted images that have been neglected due to a limited number of experts, such as fossil images, slabbed cores, or petrographic thin sections (LIMA et al., 2019), or even for environmental images (SUN et al., 2021). The ability to create distinctive models for specific sets of data allows a versatile application of those techniques.

When comparing pre-trained models, de Lima and colleagues (2019) found that both MobileNet V2 and Inception V3 showed promising results on geologic data interpretation, with MobileNet V2 having slightly better results. Also, when Sun and colleagues (2021) compared the performance of AlexNet, VGG16, ResNet50, GLNet (AlexNet), GLNet (VGG16), and GLNet (ResNet) pre-trained models on remote sensing scene classification using FE, the authors found that their proposed new model shows better results compared to other traditional DNN architectures. The proposed model GLNet (VGG16), which uses VGG16 as its base, got over 95% accuracy in analyzing a clear environment and over 94% in a cloudy environment. In contrast, the traditional VGG16 got over 93% and over 78%, for clear and cloud environments, respectively (SUN et al., 2021).

On seismic imaging classification, which is essential for oil and gas exploration, Chevitarese et al. (2018a), applied the TL of FE to train a DNN and found improvements in accuracy (CHEVITARESE et al., 2018a; CHEVITARESE et al., 2018b). Finally, Cunha et al. (2020) also trained a CNN specially developed for the fault identification problem using a dataset with synthetic patches whose seismic signal frequency content does not match that of real data. They used the first layers of this pre-trained CNN as a feature extractor to classify another dataset (CUNHA et al., 2020).

Table 3.1 presents a summary of the literature review, representing the pre-trained models used by each study identified in our review process. Table 3.2 presents the target datasets used in each study for comparing the performance of the feature extractors.

Table 3.1 provides a summary of the reviewed literature, emphasizing which pre-trained models were used for FE in each study considered in our review.

Table 3.2 summarizes the datasets in which different works evaluated different pre-trained models for FE in the literature reviewed in this work.

Table 3.1: Summary of literature review - pre-trained models

Pre-trained models	Authors																				
	Zhu et al., 2021	Xiao Qin et al., 2018	Kaur & Gandhi, 2020	Lumini & Nanni, 2019	Krishna & Kalluri, 2019	Shu, 2019	Rodrigues et al., 2018	Akilan et al., 2017	Alinsaif & Lang, 2020	Niu et al., 2020	Kumar et al., 2022	Shaha & Pawar, 2018	Nguyen et al., 2018	Pankh et al., 2019	de Lima et al., 2019	Chevtarese et al., 2018	Koeshidayatullah et al., 2020	Sun et al., 2021
InceptionV1										x								
InceptionV3		x	x	x		x		x			x		x	x	x			
InceptionResNetV2			x			x							x				x	
AlexNet			x	x	x		x	x		x		x						x
VGG-16		x	x	x	x	x		x				x		x			x	x
VFF-F																		
VGG-19		x	x	x	x	x						x						
ResNet										x								
ResNet50		x	x	x	x									x				x
ResNet101			x	x														
ResNet18	x								x									
ResNet152	x				x								x					
DenseNet				x														
DenseNet201									x									
GoogLeNet			x	x	x						x							
SqueezeNet									x		x							
MobileNet-v2									x						x			
LeNet					x													
DarkNet53											x							
DeepSea							x											
ShuffleNet											x							
Danet1																x		
Danet3																x		

Table 3.2: Summary of literature review - datasets

	Authors																			
	Zhu et al., 2021	Xiao Qin et al., 2018	Kaur & Gandhi, 2020	Lumini & Nanni, 2019	Krishna & Kalluri, 2019	Shu, 2019	Rodrigues et al., 2018	Akilan et al., 2017	Alinsaif & Lang, 2020	Niu et al., 2020	Kumar et al., 2022	Shaha & Pawar, 2018	Nguyen et al., 2018	Parikh et al., 2019	de Lima et al., 2019	Chevitarese et al., 2018	Koeshidayatullah et al., 2020	Sun et al., 2021
Target Datasets	Pallet, Defect, Data Augment ation	Oxford -17 flower, Oxford -102 flower	Clinical, Images (Figshare repository)	WHOI, ZooScan, Kaggle (subset)	CIFAR-10, Caltech Faces, MNIST, CIFAR100, ImageNet Fall2011, Flowers102, Dogs, Caltech101, Event8, 15scene, 67 Indore scene, DR, CT	Pics of dogs/ cats	LAPS, Kaggle's National Data Science Bowl, ImageNet (ISIIS)	CIFAR-10, CIFAR100, Caltech101, Caltech256, MIT67, Sun397, Pascal VOC 2012	Epistroma, Warwick-Q U, Breast Cancer Histopatho-logical, Multi-class Kather's	Trash Net	CIFAR -10	Caltech2 56, GHIM10 K	PAP-smear, 2D-Hela	RISAT-1	Microfossils, Core, Petrographic thin-sections , Rock samples	Nether-lands, Penobscot (seismic cube)	Carbonate petrographic	RSSCN7 /RSSCN 7_cloud, UC Merced/ UC Merced_ cloud

4 EXPERIMENTS

In this chapter, we discuss the experiments carried out to evaluate the different pre-trained models in different datasets. Section 4.1 presents the methodology adopted in these experiments and Section 4.2 presents and discusses the results of the experiments.

Our goal is to evaluate the performance of different available pre-trained models as feature extractors in the task of classification of geological images.

4.1 Methodology

In the experiments, we evaluated the performance of all pre-trained models presented in Section 4.2 for feature extraction in an image classification task, considering all datasets presented in Section 2.6. We evaluate the performance using the evaluation metrics presented in Section 2.5. The models presented in Section 4.2 were selected for covering a wide spectrum of the variability of architectures, including those identified in our literature review. Notice also that, since there are different versions available for each family of models, we have selected a single model for each family that presented the best overall performances according to the literature. Besides that, we adopted macro and weighted precision, recall, and f-score because we are considering in our analysis the geological dataset, which presents a reasonable imbalance.

Since we are considering 16 models and four datasets, a total of 64 experiments considering pairs of models and datasets were performed.

In each experiment, each pre-trained model was used as a feature extractor. Therefore, in this context, all initial layers (except the last one) of the model, responsible for extracting relevant features from the input images, were maintained, while the last layer was replaced by a new output layer, with N units (where N is proportional to the number of classes in the used dataset) and a linear activation functionNotice that we are using an implementation of the adopted loss function that takes as input the raw logits provided by this layer. During the model training, the weights of the initial layers (responsible for extracting features) are not adjusted (they are kept "frozen"), while the weights of the last layer are adjusted.

For each experiment, the datasets went through a homogeneous pre-processing. The pre-processing consisted of applying resizing, center cropping, and normalization. The resize is always done by decreasing or increasing the size of the smallest dimension of

the image to the size of the pre-trained model's input. Then the center crop is performed, where the central area of the image is cut so that it is square with the size of the input, and this information is used for representing the whole image. Finally, images are converted to RGB.

In addition, each model was evaluated considering a 5-fold cross-validation process. Thus, the metrics reported in Section 4.2 are averages obtained considering the performance in each test fold of this process.

Regarding the training hyperparameters, the learning rate used in this study was 0.001 with a momentum of 0.9. We adopted the Adam Optimizer with default parameters, with the Cross-Entropy[1] loss function. All executions were done using 100 epochs and early stopping with a minimal improvement of 0.001 and patience of 5.

4.2 Results

The following tables represent the model's performance according to the selected metrics for each dataset. Table 4.1 represents the model's evaluation on the Geological Images dataset. Table 4.2 represents the model's evaluation according to Stanford Cars dataset. Table 4.3 represents the model's evaluation considering the CIFAR-10 dataset. Table 4.4 represents the model's evaluation for the STL10 dataset. In each table, we highlight the model with the best performance in green and with the less performance in red.

In order to facilitate the data analysis, we have represented the data generated in our experiments in the following line charts, which demonstrate the performance (according to different metrics) of each pre-trained model for classifying the images in the four selected datasets. Figure 4.1 represents the accuracy. Figure 4.2 shows the macro precision. Figure 4.3 indicates the macro recall. Figure 4.4 demonstrates the macro f1-score. Figure 4.5 presents the weighted precision. Figure 4.6 shows the weighted recall. Figure 4.7 indicates the weighted f1-score of each model on each dataset.

The line charts in Figures 4.2-4.7 present a very similar pattern of variation of the model's performance across all datasets. We can notice also that, in general, the model's performance pattern increases and the differences among patterns decrease (resulting in a smoother pattern) in the Geological Images dataset when we consider the accuracy and the weighted averages of precision, recall, and f-score in comparison with the macro averages

[1] Notice that we are using the Pytorch implementation of cross-entropy, which takes as input raw logits.

Table 4.1: Geological Images Dataset

Geological Images Dataset							
		Macro			Weighted		
Model \Metrics	**Accuracy**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
alexnet	0,85	0,74	0,67	0,69	0,84	0,85	0,84
clip_rn50	0,93	0,86	0,83	0,84	0,92	0,93	0,92
clip_vit_b	0,93	0,86	0,83	0,84	0,93	0,93	0,93
convnext_large	0,91	0,84	0,80	0,82	0,91	0,91	0,91
densenet161	0,90	0,83	0,78	0,80	0,90	0,90	0,90
googlenet	0,87	0,75	0,72	0,73	0,86	0,87	0,86
inception_v3	0,83	0,70	0,65	0,67	0,82	0,83	0,83
mnasnet1_3	0,88	0,77	0,73	0,75	0,87	0,88	0,87
mobilenet_v3_large	0,90	0,82	0,77	0,79	0,90	0,90	0,90
regnet_y_3_2gf	0,89	0,79	0,76	0,77	0,89	0,89	0,89
resnext101_64x4d	0,88	0,79	0,74	0,76	0,88	0,88	0,88
shufflenet_v2_x2_0	0,89	0,80	0,76	0,78	0,89	0,89	0,89
squeezenet1_1	0,87	0,77	0,72	0,74	0,87	0,87	0,87
vgg19_bn	0,88	0,79	0,74	0,76	0,88	0,88	0,88
vit_h_14	0,91	0,82	0,79	0,80	0,90	0,91	0,90
wide_resnet101_2	0,89	0,79	0,75	0,77	0,89	0,89	0,89
Average	0,89	0,79	0,75	0,77	0,88	0,89	0,88
Standard Deviation	0,02	0,04	0,05	0,05	0,03	0,02	0,03

Table 4.2: Stanford Cars Dataset

Stanford Cars Dataset							
		Macro			Weighted		
Model \Metrics	**Accuracy**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	F-Score
alexnet	0,28	0,26	0,28	0,26	0,26	0,28	0,26
clip_rn50	0,82	0,82	0,82	0,82	0,82	0,82	0,82
clip_vit_b	0,83	0,83	0,83	0,83	0,83	0,83	0,83
convnext_large	0,65	0,65	0,64	0,64	0,65	0,65	0,64
densenet161	0,64	0,64	0,64	0,64	0,64	0,64	0,64
googlenet	0,41	0,41	0,41	0,41	0,40	0,41	0,40
inception_v3	0,34	0,33	0,34	0,33	0,33	0,34	0,33
mnasnet1_3	0,42	0,42	0,42	0,42	0,41	0,42	0,41
mobilenet_v3_large	0,56	0,56	0,56	0,55	0,56	0,56	0,55
regnet_y_3_2gf	0,49	0,49	0,49	0,49	0,49	0,49	0,49
resnext101_64x4d	0,35	0,35	0,35	0,34	0,34	0,35	0,34
shufflenet_v2_x2_0	0,50	0,50	0,50	0,50	0,50	0,50	0,50
squeezenet1_1	0,42	0,42	0,42	0,41	0,41	0,42	0,41
vgg19_bn	0,51	0,50	0,51	0,50	0,50	0,51	0,50
vit_h_14	0,86	0,86	0,85	0,85	0,86	0,86	0,86
wide_resnet101_2	0,44	0,44	0,44	0,44	0,44	0,44	0,44
Average	0,53	0,53	0,53	0,53	0,53	0,53	0,53
Standard Deviation	0,18	0,18	0,18	0,18	0,18	0,18	0,18

of these measures. This behavior is expected since the imbalance of this dataset is more pronounced. We can notice that, in general, the CLIP-ViT-B and VisionTransformer-H/14 models tend to show the best performances, considering all metrics in most datasets. In

Table 4.3: CIFAR-10 Dataset

CIFAR-10 Dataset							
		Macro			Weighted		
Model \Metrics	**Accuracy**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
alexnet	0,79	0,79	0,79	0,79	0,79	0,79	0,79
clip_rn50	0,88	0,88	0,88	0,88	0,88	0,88	0,88
clip_vit_b	0,95	0,95	0,95	0,95	0,95	0,95	0,95
convnext_large	0,96	0,96	0,96	0,96	0,96	0,96	0,96
densenet161	0,93	0,93	0,93	0,93	0,93	0,93	0,93
googlenet	0,87	0,87	0,87	0,87	0,87	0,87	0,87
inception_v3	0,86	0,86	0,86	0,86	0,86	0,86	0,86
mnasnet1_3	0,90	0,90	0,90	0,90	0,90	0,90	0,90
mobilenet_v3_large	0,91	0,91	0,91	0,91	0,91	0,91	0,91
regnet_y_3_2gf	0,93	0,93	0,93	0,93	0,93	0,93	0,93
resnext101_64x4d	0,95	0,95	0,95	0,95	0,95	0,95	0,95
shufflenet_v2_x2_0	0,92	0,92	0,92	0,92	0,92	0,92	0,92
squeezenet1_1	0,85	0,85	0,85	0,85	0,85	0,85	0,85
vgg19_bn	0,88	0,88	0,88	0,88	0,88	0,88	0,88
vit_h_14	0,98	0,98	0,98	0,98	0,98	0,98	0,98
wide_resnet101_2	0,95	0,95	0,95	0,95	0,95	0,95	0,95
Average	0,91	0,91	0,91	0,91	0,91	0,91	0,91
Standard Deviation	0,05	0,05	0,05	0,05	0,05	0,05	0,05

Table 4.4: STL10 Dataset

STL10 Dataset							
		Macro			Weighted		
Model \Metrics	**Accuracy**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
alexnet	0,88	0,88	0,88	0,88	0,88	0,88	0,88
clip_rn50	0,97	0,97	0,97	0,97	0,97	0,97	0,97
clip_vit_b	0,99	0,99	0,99	0,99	0,99	0,99	0,99
convnext_large	0,99	0,99	0,99	0,99	0,99	0,99	0,99
densenet161	0,98	0,98	0,98	0,98	0,98	0,98	0,98
googlenet	0,96	0,96	0,96	0,96	0,96	0,96	0,96
inception_v3	0,96	0,96	0,96	0,96	0,96	0,96	0,96
mnasnet1_3	0,97	0,97	0,97	0,97	0,97	0,97	0,97
mobilenet_v3_large	0,96	0,96	0,96	0,96	0,96	0,96	0,96
regnet_y_3_2gf	0,98	0,98	0,98	0,98	0,98	0,98	0,98
resnext101_64x4d	0,99	0,99	0,99	0,99	0,99	0,99	0,99
shufflenet_v2_x2_0	0,97	0,97	0,97	0,97	0,97	0,97	0,97
squeezenet1_1	0,91	0,91	0,91	0,91	0,91	0,91	0,91
vgg19_bn	0,96	0,96	0,96	0,96	0,96	0,96	0,96
vit_h_14	1,00	1,00	1,00	1,00	1,00	1,00	1,00
wide_resnet101_2	0,99	0,99	0,99	0,99	0,99	0,99	0,99
Average	0,97	0,97	0,97	0,97	0,97	0,97	0,97
Standard Deviation	0,03	0,03	0,03	0,03	0,03	0,03	0,03

the Geological Images dataset, the CLIP-ViT-B presents the best performance, in all metrics. The CLIP-ResNet50 also tends to perform well in the other datasets, although in the case of the CIFAR-10 and Geological Images datasets, this model's performance is rea-

Figure 4.1: Line chart representing the accuracy of each model on each dataset.

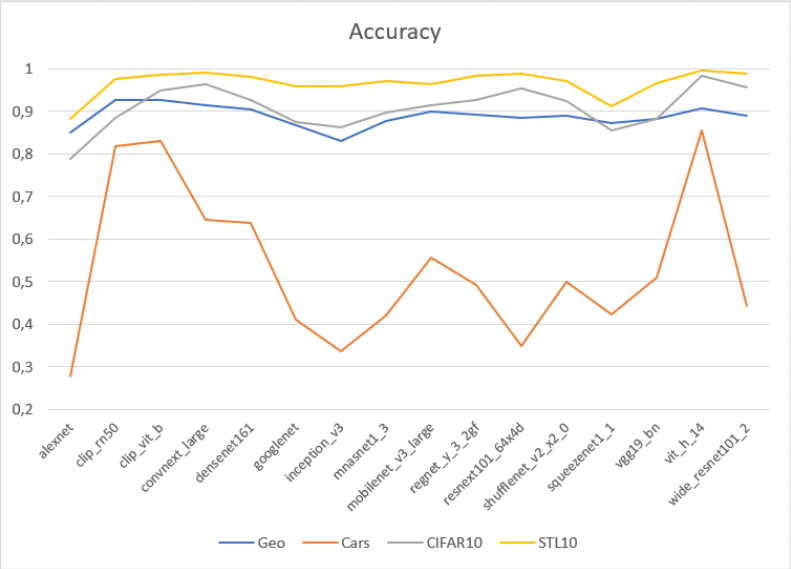


Figure 4.2: Line chart representing the macro precision of each model on each dataset.

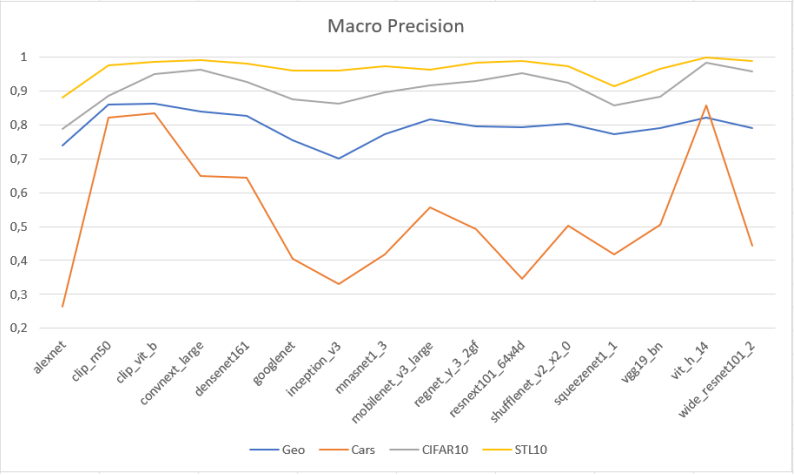
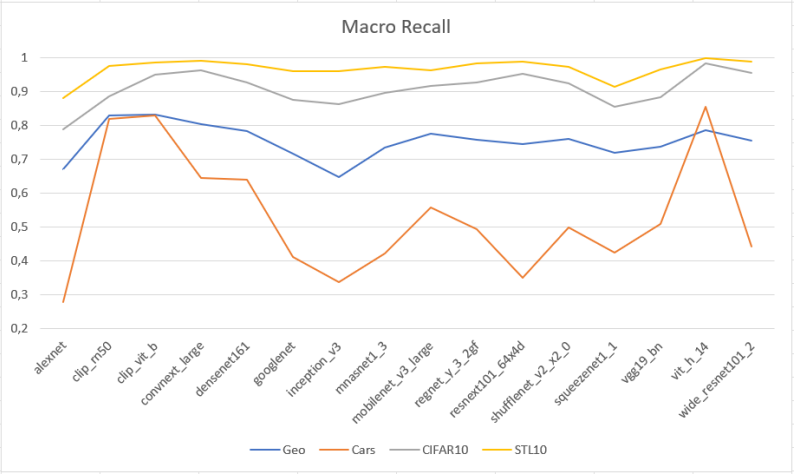


Figure 4.3: Line chart representing the macro recall of each model on each dataset.



sonably lower than the performance of CLIP-ViT-B and ViT-H/14, in all metrics. In the CIFAR-10 dataset, it is also worth highlighting the good performance of ConvNeXt Large

Figure 4.4: Line chart representing the macro f1-score of each model on each dataset.

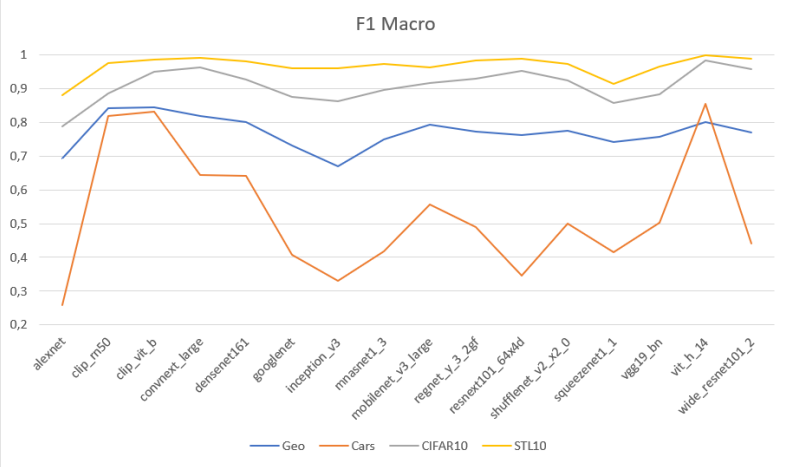


Figure 4.5: Line chart representing the weighted precision of each model on each dataset.

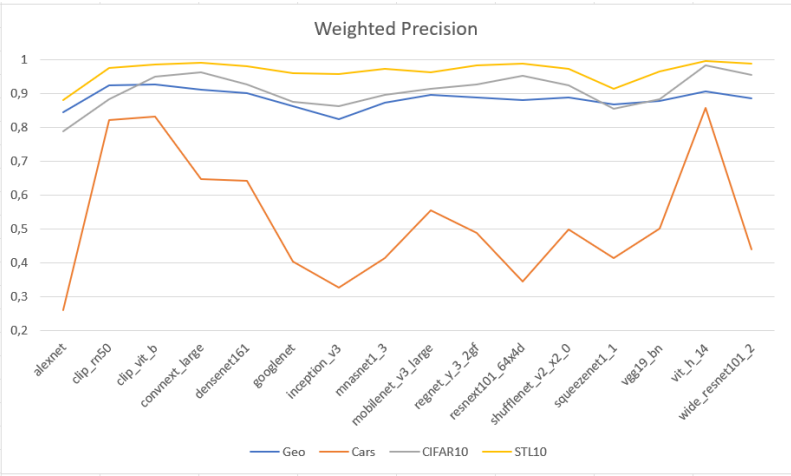
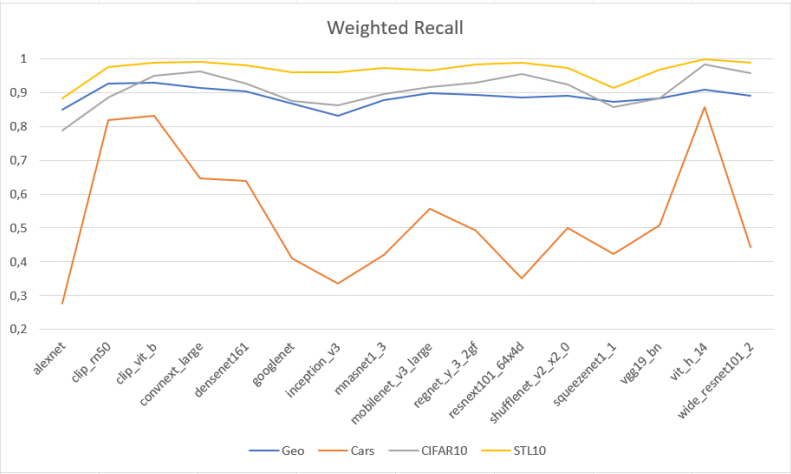
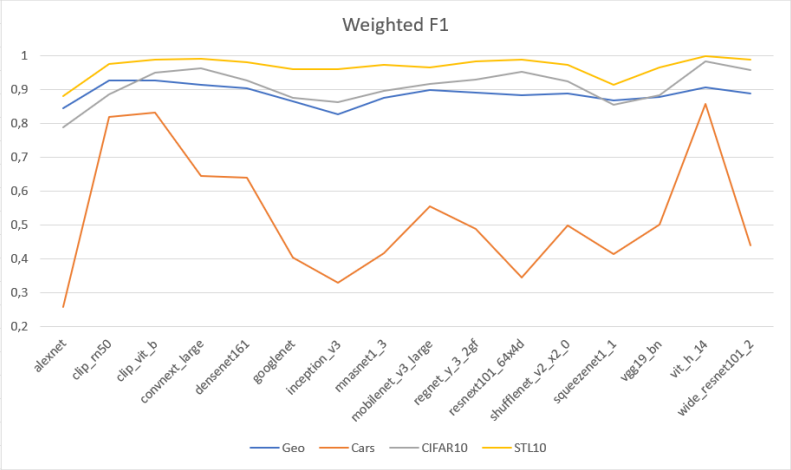


Figure 4.6: Line chart representing the weighted recall of each model on each dataset.



and ResNeXt101-64x4D. The ConvNeXt Large model also performs better than CLIP-ViT-B and CLIP-ResNet50 in STL10. In our evaluation, AlexNet presents the worst performance on most datasets, considering all metrics. Inception V3 also performed poorly on the Stanford Cars, CIFAR-10, and Geological Images datasets, where it performed the

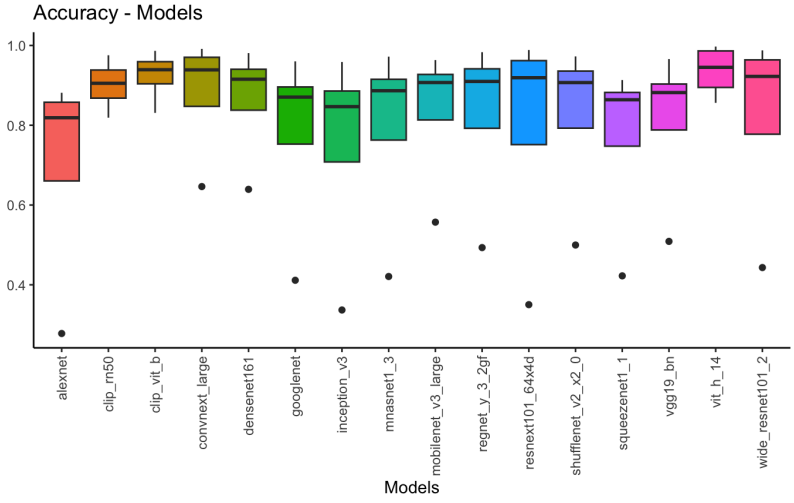
Figure 4.7: Line chart representing the weighted f1-score of each model on each dataset.



worst. Another model that had reasonably low performance compared to the others was Squezenet1-1. The poor performance of this model is more pronounced on the CIFAR-10 and STL10 datasets.

The following boxplots reveal the minimum value (excluding outliers), the maximum value (excluding outliers), the median, the first quartile, the third quartile, and outliers of the performance of the different models considering all the datasets, according to different metrics. Figure 4.8 represents the accuracy. Figure 4.9 shows the macro precision. Figure 4.10 indicates the macro recall. Figure 4.11 demonstrates the macro f1-score. Figure 4.12 presents the weighted precision. Figure 4.13 represents the weighted recall. Figure 4.14 shows the weighted f1-score of each model on each dataset. Notice that each boxplot is built from only four measures (one for each dataset), however, despite this small amount of data, this visualization can provide further insights.

Figure 4.8: Boxplot of accuracy for each model.



In the boxplots represented in Figures 4.8-4.14, it is possible to notice a pattern in

Figure 4.9: Boxplot of macro precision for each model.

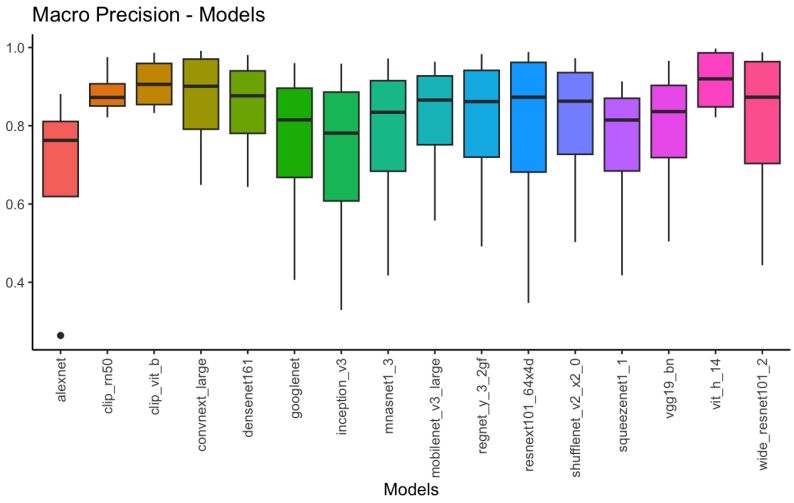


Figure 4.10: Boxplot of macro recall for each model.

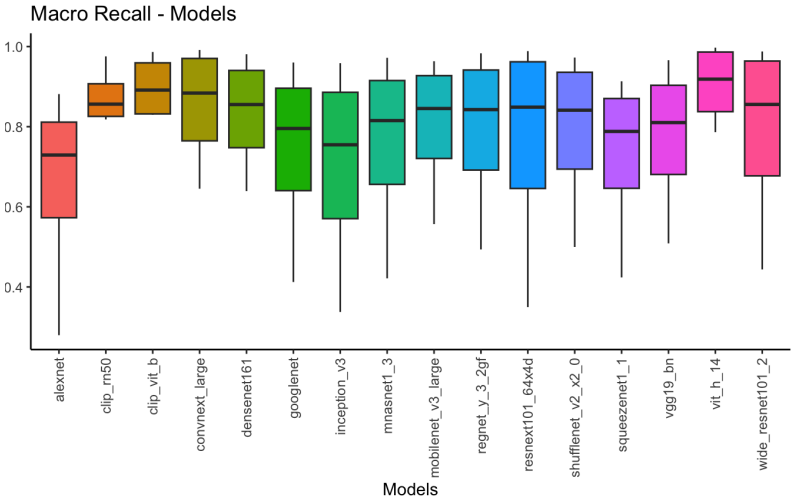
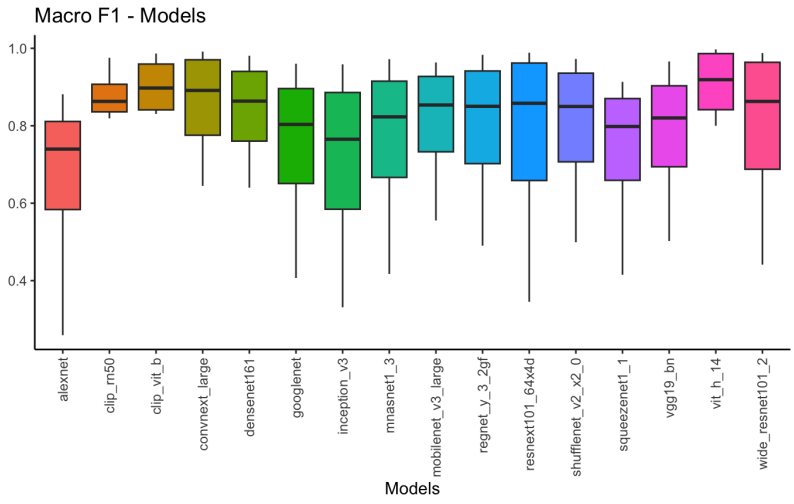


Figure 4.11: Boxplot of f1-score for each model.



the different metrics, although some differences can be noticed in the Geological Images dataset when comparing the macro averages of precision, recall, and f-score with the ac-

Figure 4.12: Boxplot of weighted precision for each model.

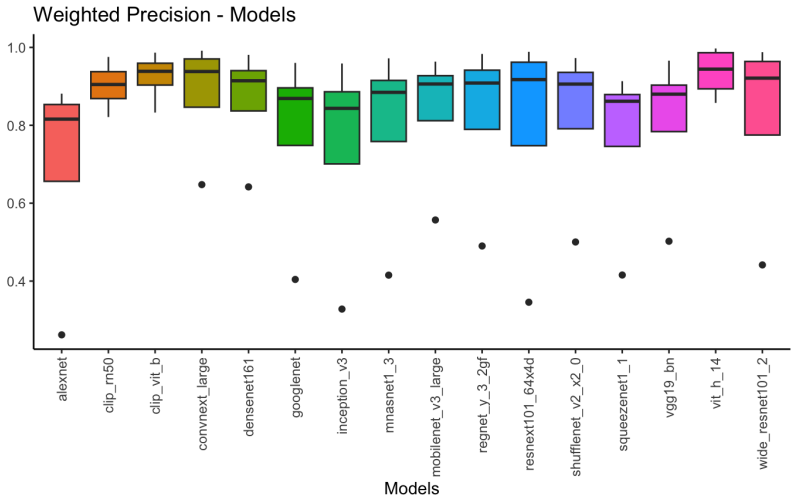


Figure 4.13: Boxplot of weighted recall for each model.

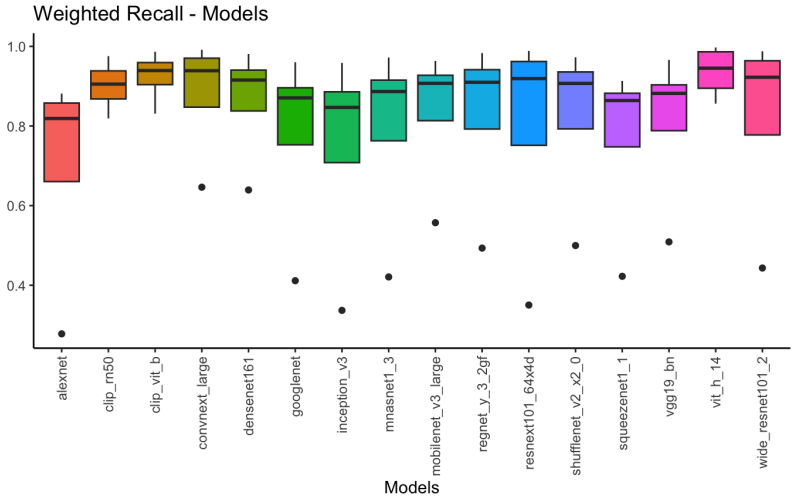
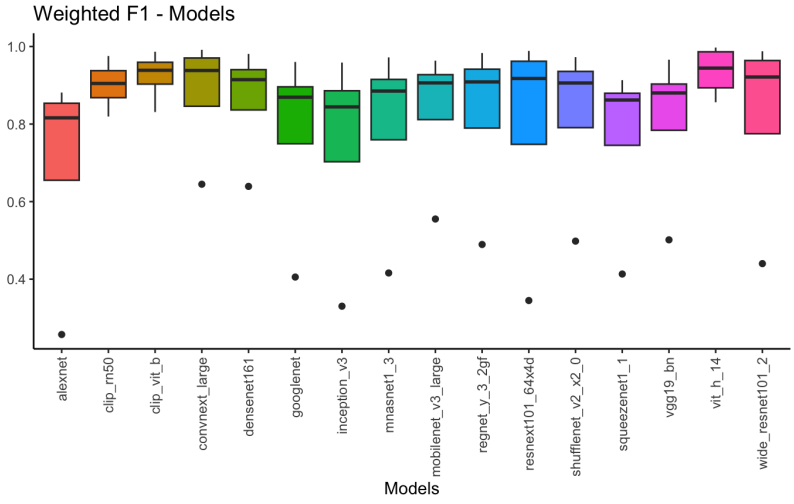


Figure 4.14: Boxplot of weighted f1-score for each model.



curacy and weighted averages of those metrics. The CLIP-ResNet50, CLIP-ViT-B, and VisionTransformer-H/14 models present the lowest variability in performance, in general.

In the macro averages of precision, recall, and f-score, CLIP-ResNet50 presents the lowest variability. On the other hand, when considering the accuracy and weighted averages of precision, recall, and f-score, CLIP-ViT-B presents less variability in its performance. It is important to note that CLIP-ResNet50, CLIP-ViT-B, and VisionTransformer-H/14 are the only models adopted in our experiments that include transformers in their architecture. It is also worth noting that, among the CNN-based architectures, ConvNeXt Large presented a higher median, compared to the other CNN-based architectures, which is similar to the medians of performances obtained by CLIP-ViT-B and VisionTransformer-H/14, although with greater variability in performance. The boxplots also suggest that, in the considered datasets, AlexNet tends to present the worst performances, in general, presenting low medians and high variability. High variability is also present in the performances of ResNeXt101-64x4D, Wide ResNet 101-2, and Inception V3, which also presents low medians when compared to the other models.

The previous analysis (Figures 4.1-4.7) suggests that some models present a very similar performance behavior across the datasets while other models exhibit behaviors that do not follow the general pattern. In order to emphasize how similar are the model's behaviors, we analyzed the Pearson correlation (COHEN et al., 2009) of the performances of each pair of models across the datasets according to all the selected metrics. The following heat maps were created to visually represent this information. In the following sequence of charts, the darker a cell gets corresponds to the lower the correlation of a given pair of models, according to a given performance metric. Figure 4.15 represents the correlation regarding the accuracy. Figure 4.16 shows the correlation regarding the macro precision. Figure 4.17 indicates the correlation regarding the macro recall. Figure 4.18 demonstrates the correlation regarding the macro f1-score. Figure 4.19 presents the correlation regarding the weighted precision. Figure 4.20 represents the correlation regarding the weighted recall. Figure 4.21 shows the correlation regarding the weighted f1-score between each pair of models.

Figures 4.15-4.21 suggest that the correlation of the performances of each pair of models presents a similar pattern in all metrics. We can notice also that, in all performance metrics, the correlation between models based solely on CNN architectures is high (in general, above 0.97). However, the performances of CLIP-ResNet50, CLIP-ViT-B, and VisionTransformer-H/14 models present a lower correlation with the performances of other models solely based on CNN. In the case of CLIP-ViT-B, the correlation with the other models is subtly lower considering accuracy and the weighted averages of precision,

Figure 4.15: Heat map representing the correlation between each pair of models regarding accuracy.

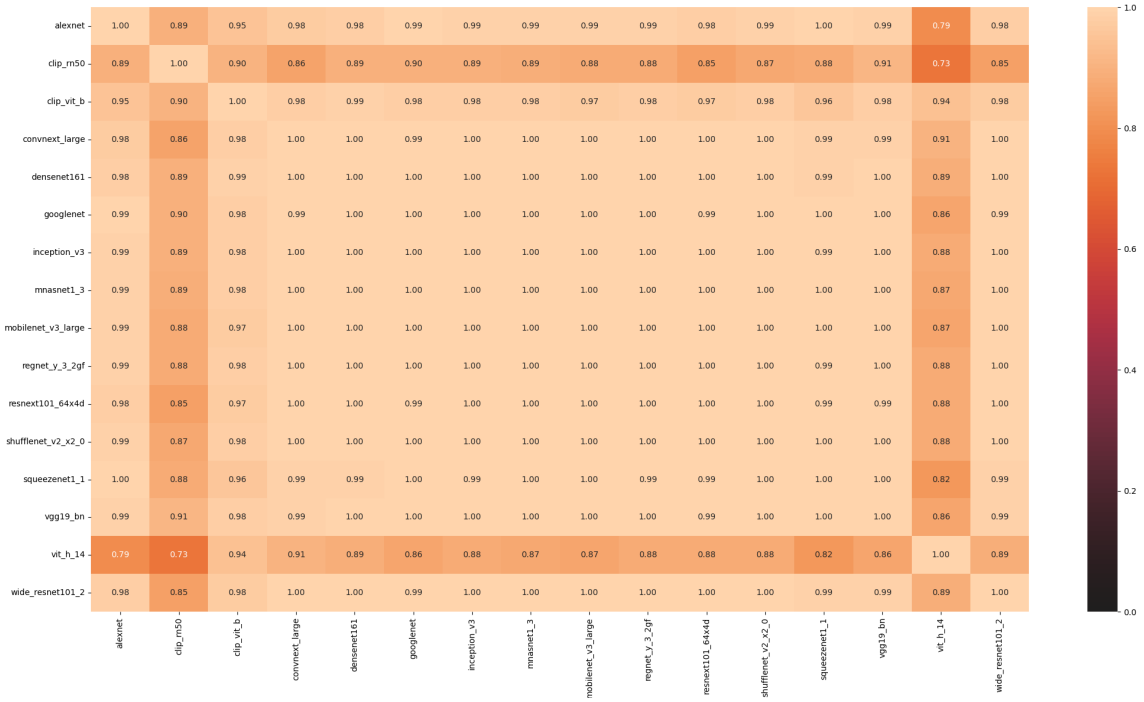
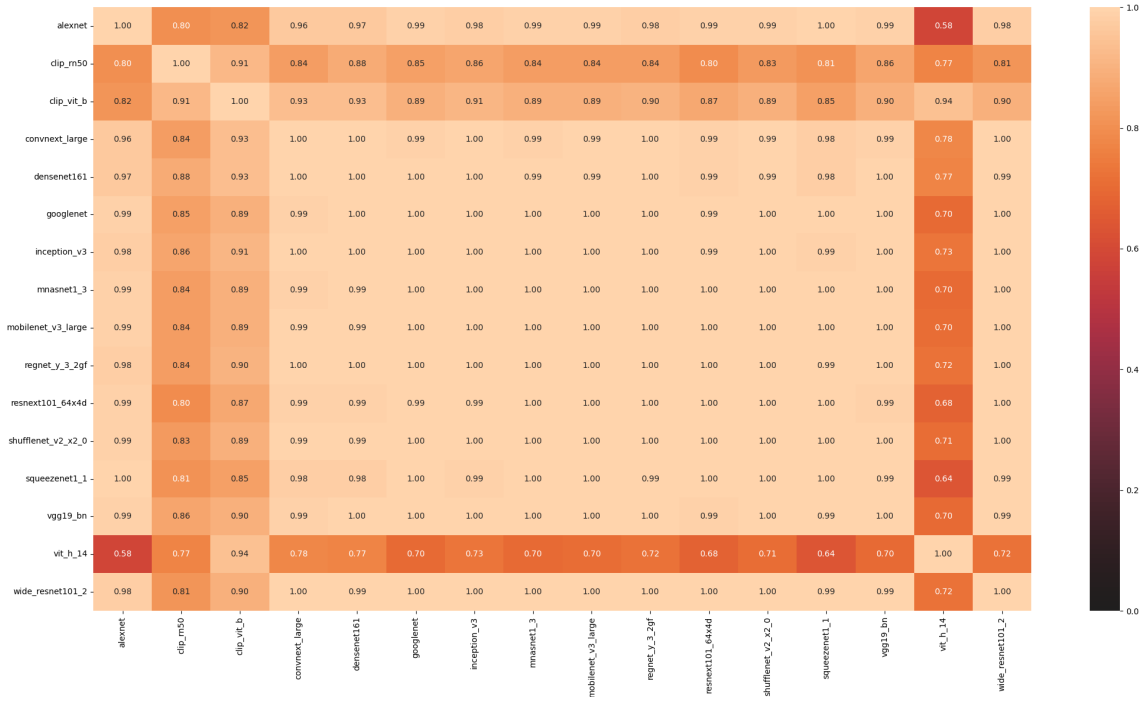


Figure 4.16: Heat map representing the correlation between each pair of models regarding the macro precision.



recall, and f-score. However, when we consider the macro averages of precision, recall, and f-score, this model's correlation is significantly lower. It is important to note that the CLIP-ResNet50, CLIP-ViT-B, and VisionTransformer-H/14 models include transformers in their architectures. This performance correlation analysis suggests that this difference

Figure 4.17: Heat map representing the correlation between each pair of models regarding the macro recall.

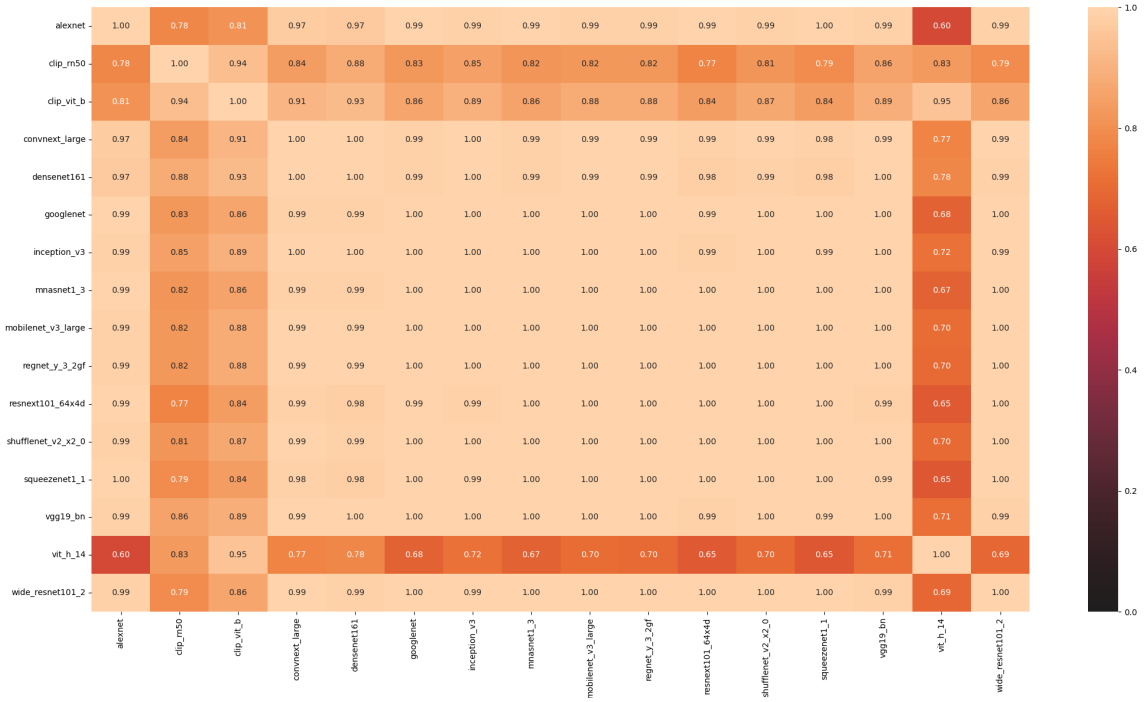
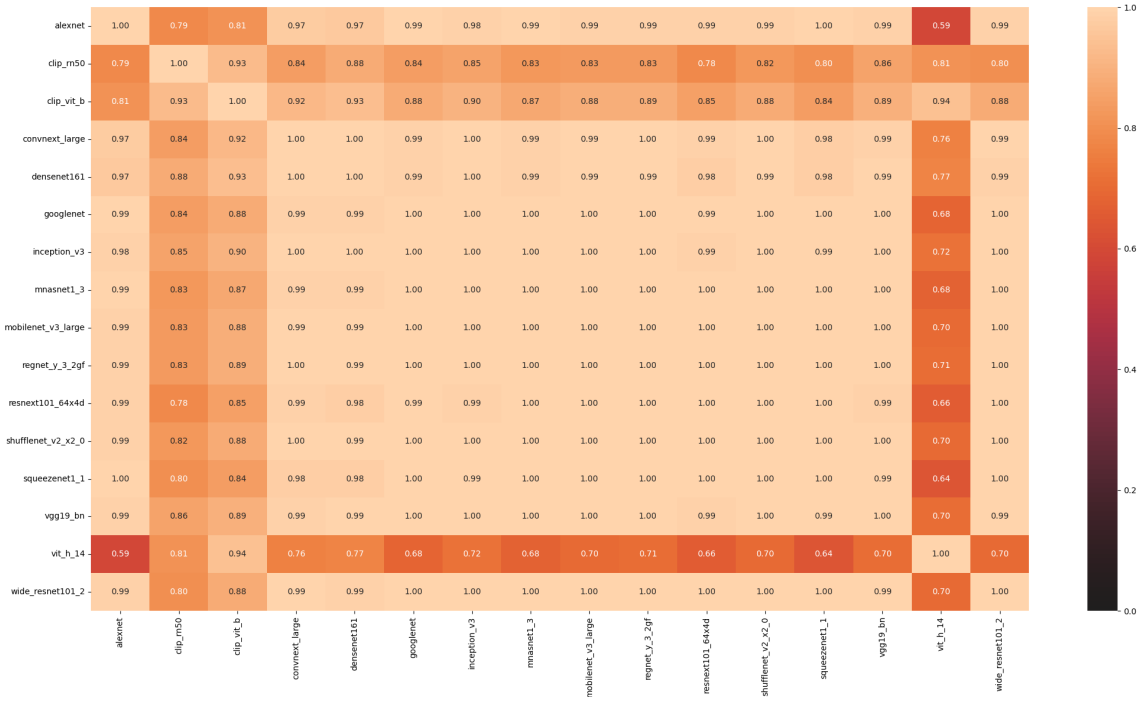


Figure 4.18: Heat map representing the correlation between each pair of models regarding the macro f1-score.



in the basic principles of the architecture of these models is correlated with this difference in the performance pattern of these models when compared to architectures based solely on CNN. Further analysis should be done in the future in order to investigate this hypothesis. The heat maps also allow us to note that the correlations among the performances

Figure 4.19: Heat map representing the correlation between each pair of models regarding the weighted precision.

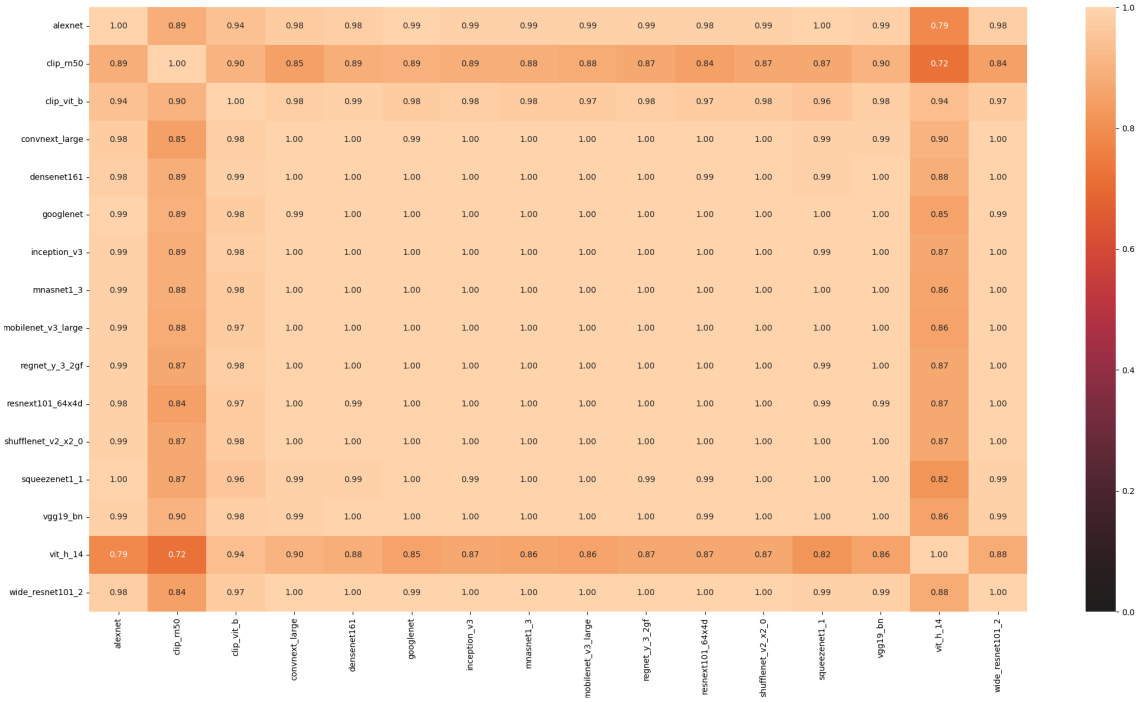
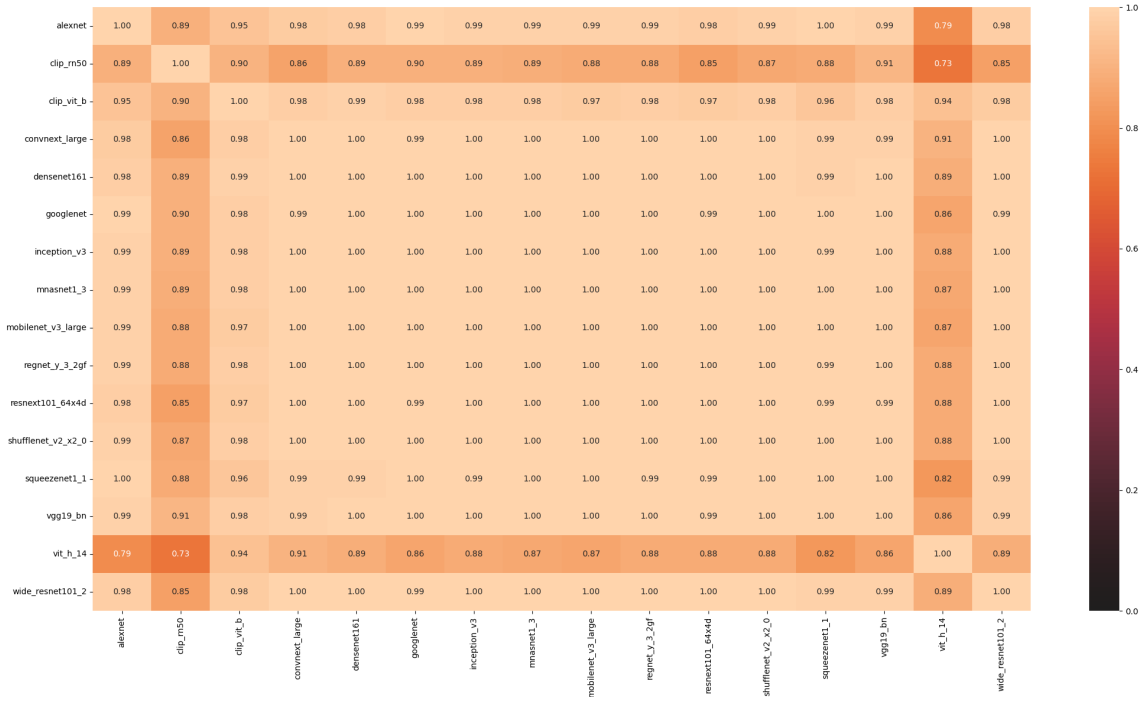


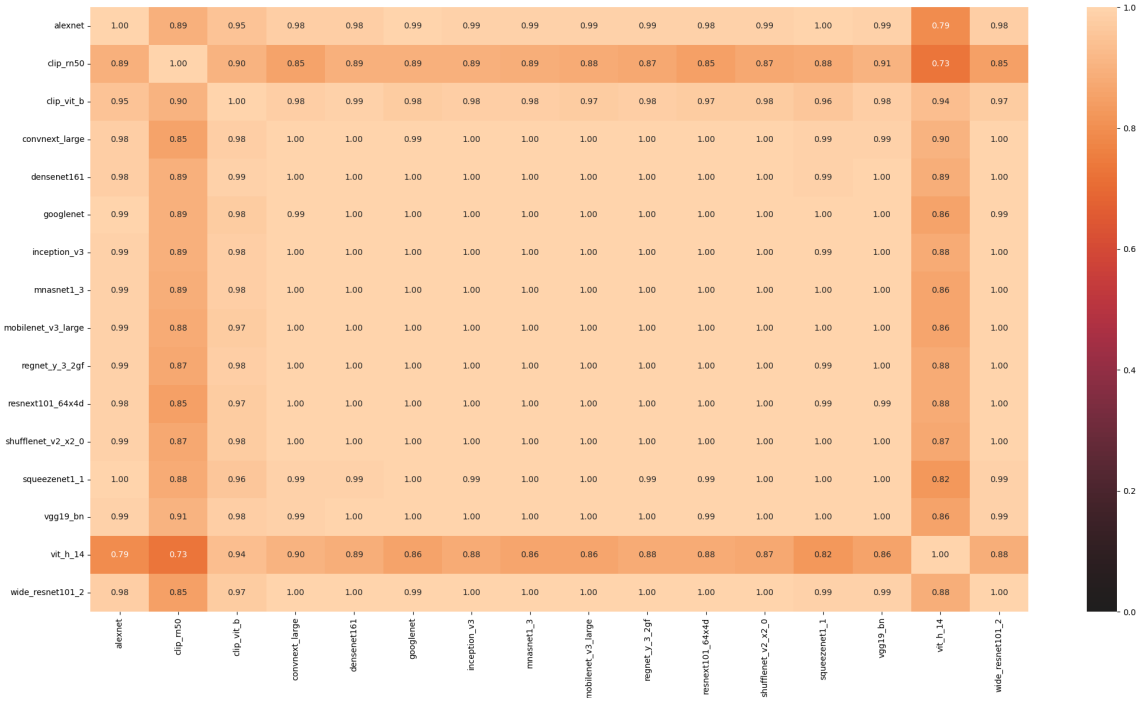
Figure 4.20: Heat map representing the correlation between each pair of models regarding the weighted recall.



of CLIP-ResNet50, CLIP-ViT-B, and VisionTransformer-H/14 models are not high when compared with the correlations among the performances of models based solely on CNN.

In the previous analyses, we focused on the performance of the models considered in our experiments. In the following charts, we focused on analyzing the datasets consid-

Figure 4.21: Heat map representing the correlation between each pair of models regarding the weighted f1-score.



ered in our experiments. Each chart represents how the performances of all models vary across the different datasets, considering different metrics. This information is already present in Figures 4.1 - 4.7. However, these charts provide an alternative visualization of this information that can suggest, for example, how hard is to classify each dataset. Figure 4.22 focuses on accuracy. Figure 4.2 represents macro precision. Figure 4.24 shows macro recall. Figure 4.25 indicates macro f1-score. Figure 4.26 demonstrates weighted precision. Figure 4.27 presents weighted recall. Figure 4.28 focuses on the weighted f1-score of all models across the datasets.

Figure 4.22: Chart representing the accuracy of all models on different datasets.

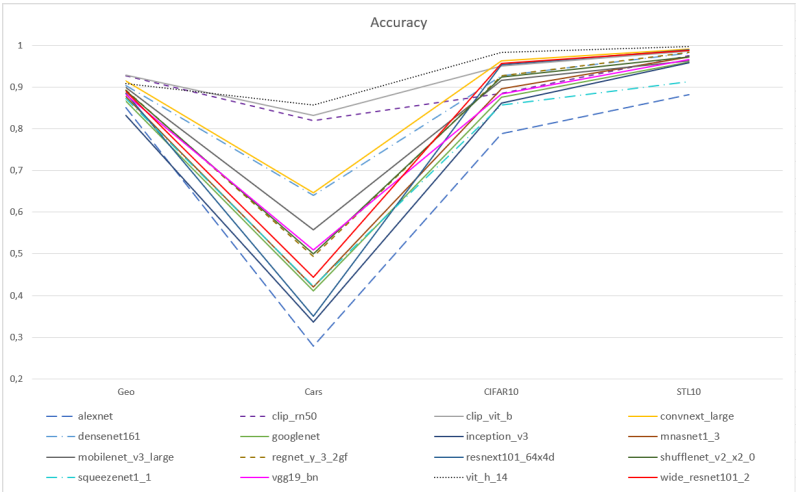


Figure 4.23: Chart representing the macro precision of all models on different datasets.

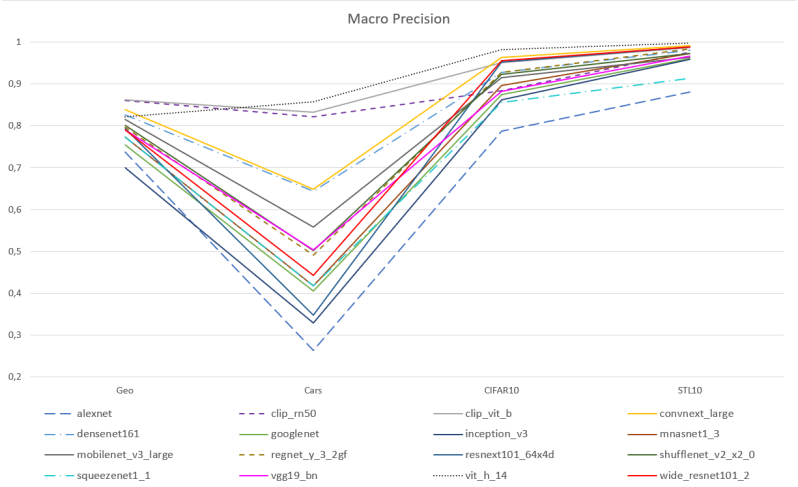


Figure 4.24: Chart representing the macro recall of all models on different datasets.

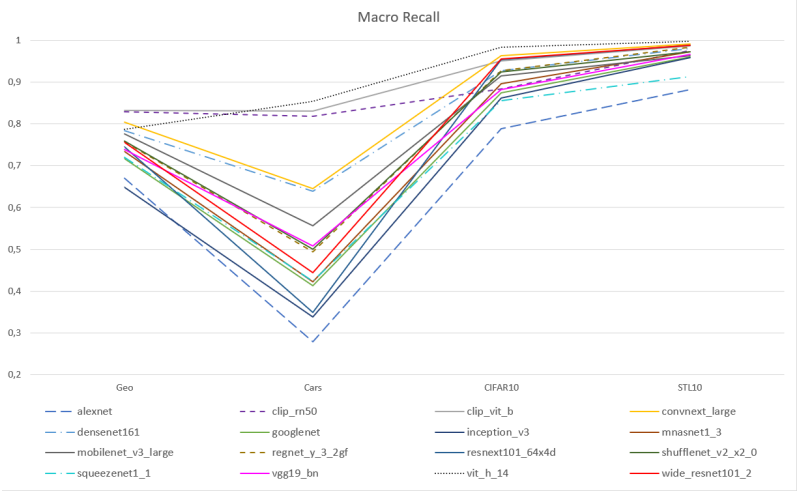
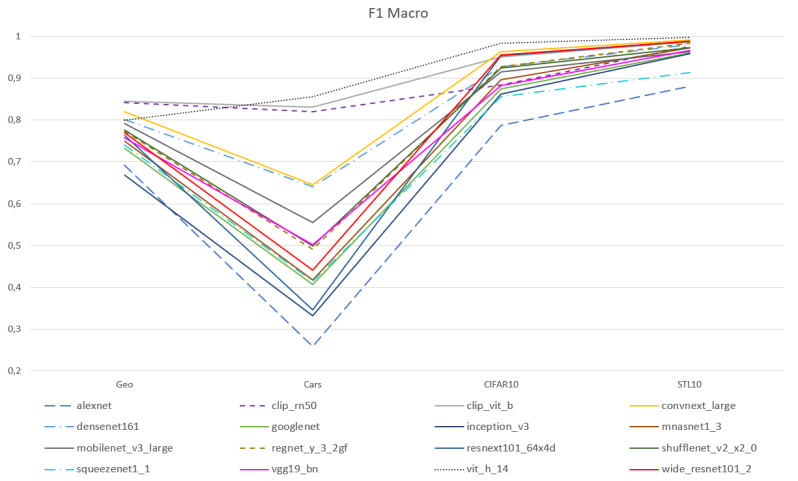


Figure 4.25: Chart representing the macro f1-score of all models on different datasets.



The line charts represented in Figures 4.22-4.28 present a similar pattern that can be seen across the different metrics. There are subtle differences when comparing the macro averages of precision, recall, and f-score with accuracy and weighted averages of

Figure 4.26: Chart representing the weighted precision of all models on different datasets.

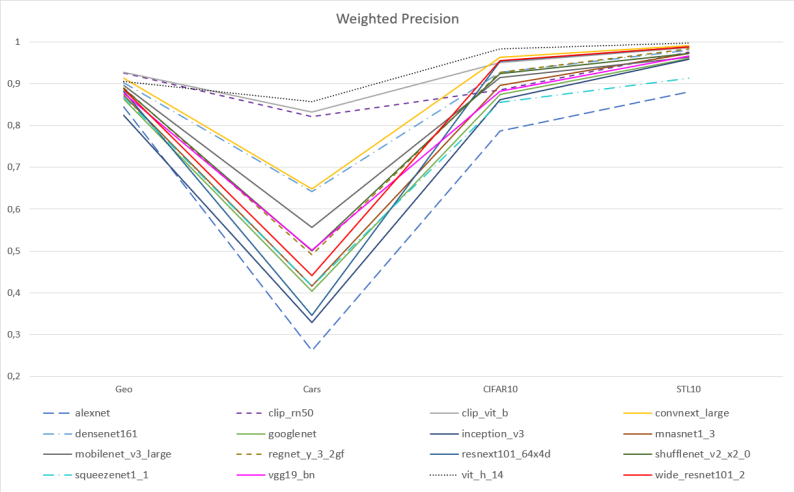


Figure 4.27: Chart representing the weighted recall of all models on different datasets.

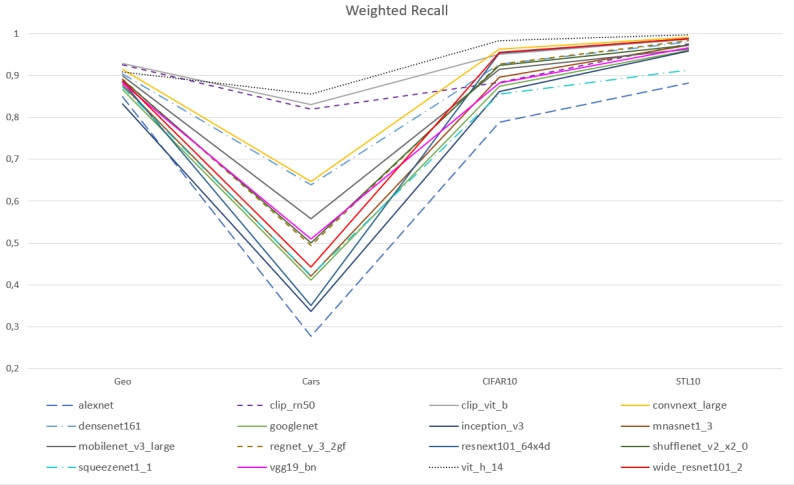
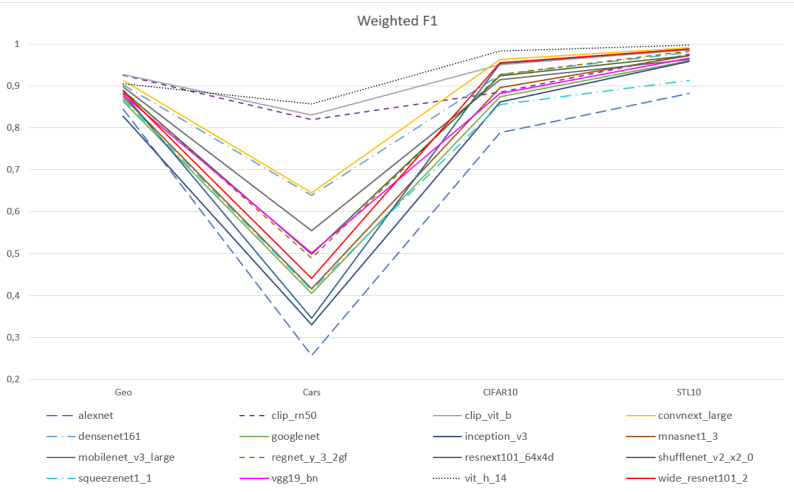


Figure 4.28: Chart representing the weighted f1-score of all models on different datasets.



those metrics. Note that, in general, the models tend to perform better in the STL10 dataset, in second place CIFAR-10 has the best overall results, in third place the dataset of Geological Images and, finally, the dataset with the worst performances, in general,

is the Stanford Cars. This is an expected result since this dataset has a large number of classes, few samples per class, and the inclusion of images with different sizes and features at different scales. The Geological Images dataset has similar properties but has fewer classes and more samples per class than Stanford Cars, although it presents a greater imbalance.

The following boxplots represent the performances considering all the models in each dataset, according to different metrics. Figure 4.29 represents the variation of accuracy. Figure 4.30 shows the variation of macro precision. Figure 4.31 indicates the variation of macro recall. Figure 4.32 demonstrates the variation of macro f1-score. Figure 4.33 presents the variation of weighted precision. Figure 4.34 represents the variation of weighted recall. Figure 4.35 shows the variation of weighted f1-score in each dataset.

Figure 4.29: Boxplot of accuracy for each dataset.

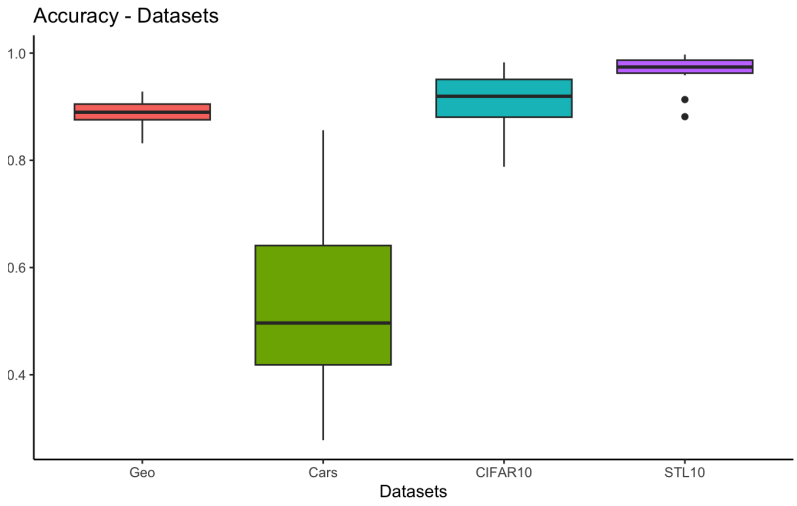


Figure 4.30: Boxplot of macro precision for each dataset.

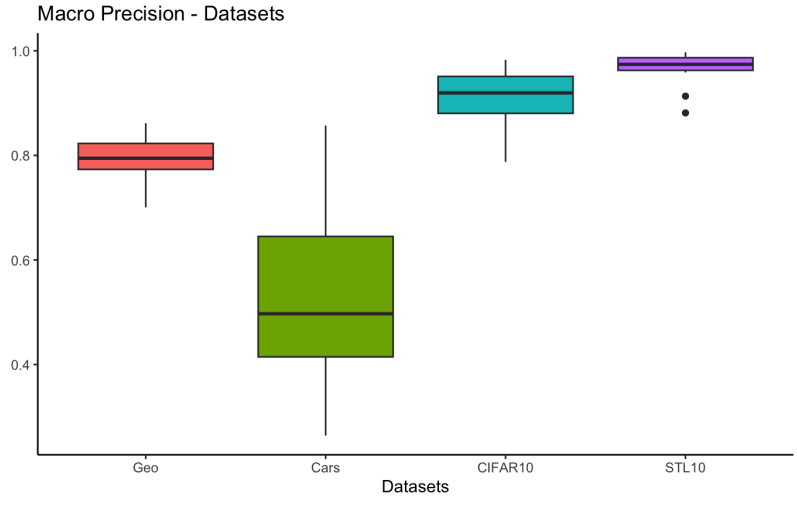


Figure 4.31: Boxplot of macro recall for each dataset.

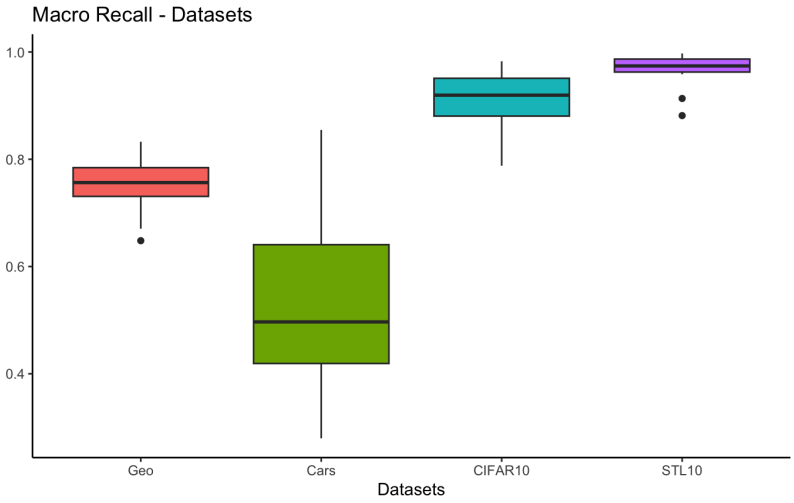


Figure 4.32: Boxplot of macro f1-score for each dataset.

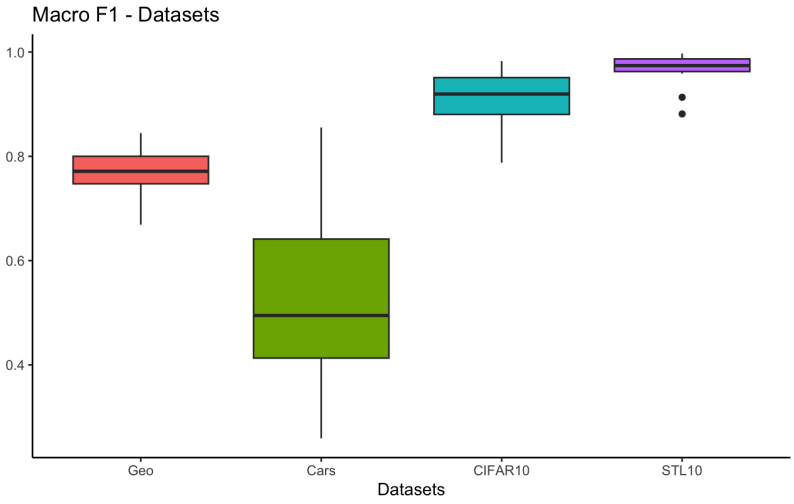
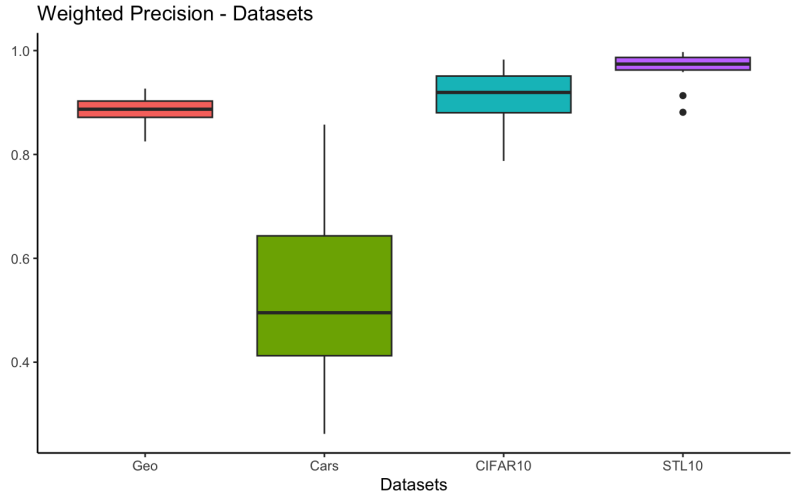


Figure 4.33: Boxplot of weighted precision for each dataset.



The boxplots represented in Figures 4.29-4.35 present a similar pattern across the different metrics. There are subtle differences when comparing the macro averages of

Figure 4.34: Boxplot of weighted recall for each dataset.

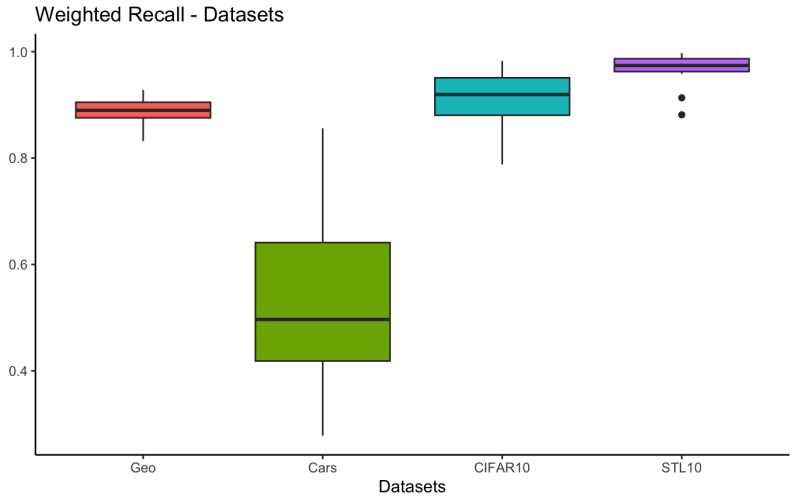
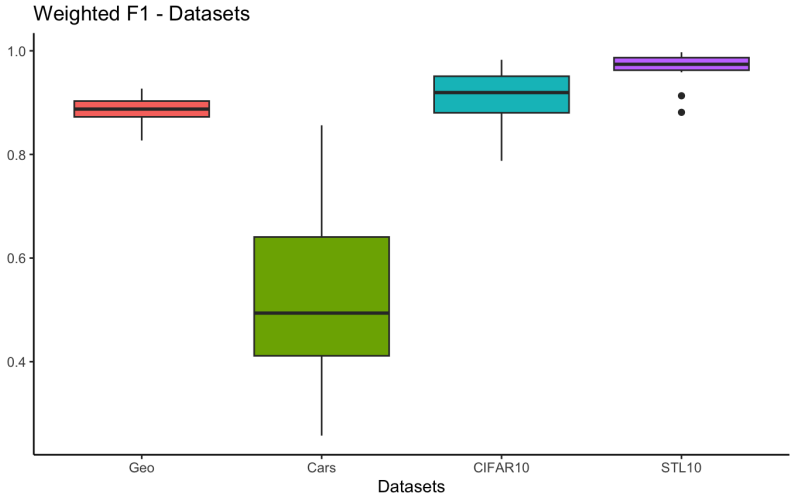


Figure 4.35: Boxplot of weighted f1-score for each dataset.



precision, recall, and f-score with accuracy and the weighted averages of those metrics. These boxplots emphasize some properties found in the previous graphs (Figures 4.22 - 4.28). Firstly, these charts provide evidence that the Stanford Cars dataset is the most challenging among those analyzed, with the worst performances and the greatest variability of performances in all metrics. Besides that, we can notice also that the STL10 dataset and Geological Images have a smaller variability in the performance of the different models when compared with the other two datasets.

5 CONCLUSIONS

In this work, our goal was to compare the performance of sixteen pre-trained neural networks for FE in four different datasets, with an emphasis on geological image classification.

By analyzing the accuracy and macro and weighted averages of f-score, recall, and precision, our experiments have shown that CLIP-ViT-B achieved the best results for the Geological Images dataset, which was the main focus of this work. Our experiments showed also that CLIP-ResNet50 and VisionTransformer-H/14 also achieved similar performances.

When we focus on the general performances, considering all the datasets, our experiments suggest that CLIP-ViT-B and VisionTransformer-H/14 achieved better performance results for all datasets and low variability in their performances. Besides that, CLIP-ResNet50 achieved performance similar to the performance achieved by CLIP-ViT-B, and VisionTransformer-H/14 and even lower variability. It is important to notice that CLIP-ViT-B, VisionTransformer-H/14, and CLIP-ResNet50 include transformers in their architectures. Thus, our results suggest that the principles underlying the transformers can be the reason corroborating these remarkable results, but further studies should be carried out to investigate this hypothesis.

Among the CNN-based architectures, ConvNeXt Large presents the best performance, in general, and lower variability when compared to other CNN-based architectures. AlexNet showed the worst performance and high variability. Besides that, ResNeXt101-64x4D, Wide ResNet 101-2, and Inception V3 also showed high variability.

Our analysis also showed that the performances of models based solely on CNN architectures present a high Pearson correlation in all performance metrics. However, the performances of CLIP-ResNet50, CLIP-ViT-B, and VisionTransformer-H/14 models show a lower correlation with other models based solely on CNN. This difference in performance is hypothesized to be due to differences regarding the basic principles of the architecture of these models. However, the correlations among the performances of CLIP-ResNet50, CLIP-ViT-B, and VisionTransformer-H/14 models are not high, if compared to correlations among the performances of CNN-based models. Further studies are needed to better investigate this finding.

Our analysis also has shown that the selected models performed better on the STL10 dataset, followed by CIFAR-10, then the Geological Images dataset, and finally

the Stanford Cars dataset. Thus, we can conclude that the Stanford Cars dataset is the most challenging dataset evaluated in this work. The Stanford Cars present a particularly large image size (when compared with CIFAR-10 and STL10, for example) and a high amount of classes with just a few samples per class. These characteristics may explain this result. The Geological Images dataset shares some of the properties of the Stanford Cars dataset, but it presents a higher imbalance, has a lower amount of classes, and has more images per class, in general.

The investigation presented in this work can provide evidence that supports the choice of models for transfer learning in image classification tasks involving the Geological Images dataset. Since our evaluation also covered other image datasets, it can also suggest reasonable choices for transfer learning in other domains.

In future works, to make the analysis more comprehensive, it is important to expand it by including other image datasets. Besides that, the investigation can also be expanded to include more pre-trained models that eventually were not considered in the scope of this work. Furthermore, future works could also investigate the relationship between the underlying principles of each architecture, the properties of the datasets used in the pre-training of these models, and the properties of the target datasets, in which the pre-trained models are applied to extract features. This investigation can reveal insights into what makes the pre-trained model best suited for each task.

REFERENCES

ABEL, M. et al. A knowledge organization system for image classification and retrieval in petroleum exploration domain. In: **Proceedings of ONTOBRAS**. [S.l.: s.n.], 2019.

ALZUBAIDI, L. et al. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. **Journal of big Data**, v. 8, n. 1, p. 1–74, 2021.

ARSLAN, Y. et al. A comparison of pre-trained language models for multi-class text classification in the financial domain. In: **Companion Proceedings of the Web Conference 2021**. [S.l.: s.n.], 2021. p. 260–268.

BAKER, N. A.; ZENGELER, N.; HANDMANN. Uwe. a transfer learning evaluation of deep neural networks for image classification. **Machine Learning and Knowledge Extraction**, v. 4, n. 1, p. 22–41, 2022.

BORG, M. et al. Safely entering the deep: A review of verification and validation for machine learning and a challenge elicitation in the automotive industry. arxiv. Preprint. 2018.

CADAVID, U.; PABLO, J. et al. Machine learning applied in production planning and control: a state-of-the-art in the era of industry 4.0. **Journal of Intelligent Manufacturing**, v. 31, n. 6, p. 1531–1558, 2020.

CHEVITARESE, D. et al. **Transfer learning applied to seismic images classification**. [S.l.]: AAPG Annual and Exhibition, 2018.

CHEVITARESE, D. S. et al. Efficient classification of seismic textures. In: IEEE. **2018 International Joint Conference on Neural Networks (IJCNN)**. [S.l.], 2018. p. 1–8.

COATES, A.; NG, A.; LEE, H. An analysis of single-layer networks in unsupervised feature learning. In: JMLR WORKSHOP AND CONFERENCE PROCEEDINGS. **Proceedings of the fourteenth international conference on artificial intelligence and statistics**. [S.l.], 2011. p. 215–223.

COHEN, I. et al. Pearson correlation coefficient. **Noise reduction in speech processing**, Springer, p. 1–4, 2009.

CUNHA, A. et al. Seismic fault detection in real data using transfer learning from a convolutional neural network pre-trained with synthetic seismic data. **Computers & Geosciences**, v. 135, n. 10434, p. 4, 2020.

DARA, S.; TUMMA, P. Feature extraction by using deep learning: A survey. In: **2018 Second International Conference on Electronics**. [S.l.]: Communication and Aerospace Technology (ICECA). IEEE p, 2018. p. 1795–1801.

DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: IEEE. **2009 IEEE conference on computer vision and pattern recognition**. [S.l.], 2009. p. 248–255.

DOSOVITSKIY, A. et al. An image is worth 16x16 words: Transformers for image recognition at scale. **arXiv preprint arXiv:2010.11929**, 2020.

FAWAZ, H. I. et al. Transfer learning for time series classification. In: **2018 IEEE international conference on big data (Big Data)**. [S.l.]: IEEE p, 2018. p. 1367–1376.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S.l.]: MIT press, 2016.

HE, K. et al. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition p**. [S.l.: s.n.], 2016. p. 770–778.

HOLLINK, L. et al. Semantic annotation of image collections. In: **Knowledge capture**. [S.l.: s.n.], 2003. v. 2.

HOWARD, A. et al. Searching for mobilenetv3. In: **Proceedings of the IEEE/CVF international conference on computer vision**. [S.l.: s.n.], 2019. p. 1314–1324.

HUANG, G. et al. Densely connected convolutional networks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2017. p. 4700–4708.

IANDOLA, F. N. et al. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. **arXiv preprint arXiv:1602.07360**, 2016.

JOVEL, J.; GREINER, R. An introduction to machine learning approaches for biomedical research. **Frontiers in Medicine**, v. 8, 2021.

KARPATNE, A. et al. Machine learning for the geosciences: Challenges and opportunities. **IEEE Transactions on Knowledge and Data Engineering**, v. 31, n. 8, p. 1544–1554, 2018.

KRAUSE, J. et al. 3d object representations for fine-grained categorization. In: **4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)**. Sydney, Australia: [s.n.], 2013.

KRIZHEVSKY, A. One weird trick for parallelizing convolutional neural networks. **arXiv preprint arXiv:1404.5997**, 2014.

KRIZHEVSKY, A.; HINTON, G. et al. Learning multiple layers of features from tiny images. Toronto, ON, Canada, 2009.

KUMAR, J. S.; ANUAR, S.; HASSAN, N. H. Transfer learning based performance comparison of the pre-trained deep neural networks. **International Journal of Advanced Computer Science and Applications**, v. 13, p. 1, 2022.

LANCHANTIN, J. et al. General multi-label image classification with transformers. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2021. p. 16478–16488.

LECUN, B.; HINTON, G. Deep learning. **Nature.**, v. 521, p. 436–44, 2015.

LI, Z.; HOIEM, D. Learning without forgetting. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 40, n. 12, p. 2935–2947, 2017.

LIMA, R. P. D. et al. Deep convolutional neural networks as a geological image classification tool. **The Sedimentary Record**, v. 17, n. 2, p. 4–9, 2019.

LIU, Z. et al. A convnet for the 2020s. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2022. p. 11976–11986.

LUMINI, A.; NANNI, L. Deep learning and transfer learning features for plankton classification. **Ecological informatics**, v. 51, p. 33–43, 2019.

MA, N. et al. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: **Proceedings of the European conference on computer vision (ECCV)**. [S.l.: s.n.], 2018. p. 116–131.

MALLOUH, A. A.; QAWAQNEH, Z.; BARKANA, B. D. Utilizing cnns and transfer learning of pre-trained models for age range classification from unconstrained face images. **Image and Vision Computing**, Elsevier, v. 88, p. 41–51, 2019.

MANIAR, H. et al. Machine-learning methods in geoscience. In: **2018 SEG International Exposition and Annual Meeting**. [S.l.]: OnePetro, 2018.

MITCHELL, T. **Machine Learning**. McGraw-Hill, 1997. (McGraw-Hill International Editions). ISBN 9780071154673. Available from Internet: <https://books.google.pt/books?id=EoYBngEACAAJ>.

PFERD, J. The challenges of integrating structured and unstructured data. In: **14th Petroleum Network Education Conference**. S.l.: s.n, 2010.

RADFORD, A. et al. Learning transferable visual models from natural language supervision. In: PMLR. **International conference on machine learning**. [S.l.], 2021. p. 8748–8763.

RADOSAVOVIC, I. et al. Designing network design spaces. In: **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**. [S.l.: s.n.], 2020. p. 10428–10436.

RUSSELL, S. J. **Artificial intelligence a modern approach**. [S.l.]: Pearson Education, Inc., 2010.

SHRESTHA, A.; MAHMOOD, A. Review of deep learning algorithms and architectures. **IEEE access**, v. 7, p. 53040–53065, 2019.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

SUN, H. et al. Convolutional neural networks based remote sensing scene classification under clear and cloudy environments. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision p**. [S.l.: s.n.], 2021. p. 713–720.

SZEGEDY, C. et al. Going deeper with convolutions. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2015. p. 1–9.

SZEGEDY, C. et al. Rethinking the inception architecture for computer vision. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 2818–2826.

TAN, M. et al. Mnasnet: Platform-aware neural architecture search for mobile. In: **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**. [S.l.: s.n.], 2019. p. 2820–2828.

THOMEE, B. et al. Yfcc100m: The new data in multimedia research. **Communications of the ACM**, ACM New York, NY, USA, v. 59, n. 2, p. 64–73, 2016.

TODESCATO, M. V. et al. Multiscale context features for geological image classification. In: **Proceedings of the 25th International Conference on Enterprise Information Systems (ICEIS)**. [S.l.: s.n.], 2023.

VANI, S.; RAO, T. V. M. An experimental approach towards the performance assessment of various optimizers on convolutional neural network. In: **2019 3rd international conference on trends in electronics and informatics (ICOEI)**. [S.l.]: IEEE p, 2019. p. 331–336.

VASWANI, A. et al. Attention is all you need. **Advances in neural information processing systems**, v. 30, 2017.

XIE, S. et al. Aggregated residual transformations for deep neural networks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2017. p. 1492–1500.

YAMASHITA, R. et al. Convolutional neural networks: an overview and application in radiology. **Insights into imaging**, v. 9, n. 4, p. 611–629, 2018.

ZAGORUYKO, S.; KOMODAKIS, N. Wide residual networks. **arXiv preprint arXiv:1605.07146**, 2016.

ZHUANG, F. et al. A comprehensive survey on transfer learning. In: **Proceedings of the IEEE 109**. 1: [s.n.], 2020. p. 43–76.