Universidade de Aveiro
Departamento de Electrónica Telecomunicações e Informàtica

Fundamentos de Aprendizagem Automática
2^{nd} project

Deep learning
Landscape recognition

Authors:
Stanisław Franczyk, 112059
Stefano Petrina, 112056

Course instructor:
Petia Georgieva

Keywords: Deep learning, Image recognition, Convolutional Neural Network

January 20, 2023

Contents

1 Introduction

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain allowing it to "learn" from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.

1.1 Chosen data set

For the second project of the subject Foundation of Machine Learning the set selected is **Landscape Recognition - image dataset**[1] from *kaggle* dataset.

1.2 Project goal

The goal of this project is to do a review and compare different articles and works related to image recognition and to choose the best method to solve the problem. Using these informations, next step is to create a program that is able to recognize the type of landscape (the choice is between five different types of landscape: coast, desert, forest, glacier and mountains) using a deep learning model. At the end the accuracy of the project will be tested using new landscape pictures.

2 State-of-the-art review

2.1 A Review of Deep Learning in Image Recognition

The research paper describes a few selected deep-learning approaches which have been used in the field of image classification. Each of them is based on a Convolutional neural network(CNN). These are AlexNet, VGG, GoogLeNet and ResNet [2].

2.2 Choice of the model

2.2.1 Trained vs Pre-Trained Models

In 2017 the Institute of Electrical and Electronics Engineers (IEEE) published an article comparing the results obtained using training and a pre-trained convolutional neural network for an image classification problem (in that case it was about histopathology image classification). The results show that employing a pre-trained network may be the better option. [3]

2.2.2 Convolutional Neural Network using Keras

In the area of image classification, the best technique to use is the CNN (Convolutional Neural Network). To apply a CNN in this project we used the open-source software library Keras, which is a popular API for neural networks written in Python. [4]

2.3 Landscape classification with deep neural networks

In this article authors describe their way to train model to recognizing ad clasification of landscapes. They used DCNN and pre-trained DCNN model to do it. Their model was more complex than our, it can recognize individual parts of landscape. The model can be used to analysis and interpretation of geomorphic processes [5].

3 Data description

The dataset consists of 5 different classes. Each class represents a kind of landscape. The classes are:

1. Coast - *images belonging to coastal areas, or simply beaches;*
2. Desert - *images of desert areas such as Sahara Thar, etc;*
3. Forest - *images belonging to forest areas such as Amazon;*
4. Glacier - *images of glaciers, for example the Antarctic;*
5. Mountains - *images of the mountain areas such as the Himalayas.*

This data is divided into 3 sub categories: training, validation and testing data directories.

Training Dataset: The sample of data used to fit the model. The model sees and learns from this data.

Validation Dataset: The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.

Test Dataset: The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

3.1 Class Distribution

The fist first thing to do is to see how the classes are distributed in the dataset. To do so we created the pie charts of the three classes using the function *pie* of the *plotly* library.

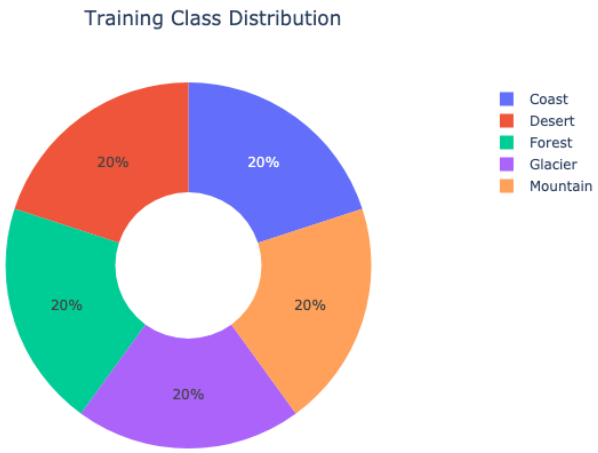

Figure 1: Training Class Distribution

Figure 1 represents the distribution of the training dataset. As we can see, all the

classes are equally distributed. This ensure that the model cannot be biased towards any class. The same result is obtained with the other two datasets.

3.2 Data Visualization

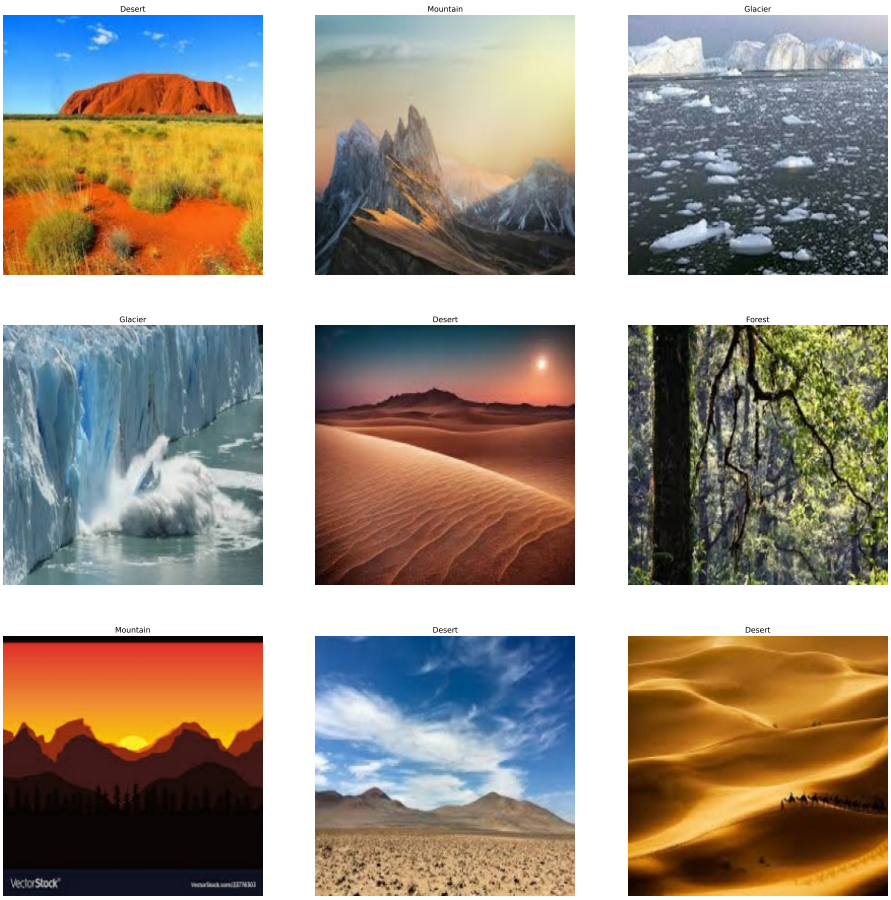

Figure 2: Training dataset pictures

Figure 2 shows nine pictures belonging to the **Training Dataset**. The other two datasets contain similar images. From the images we can clearly notice that the model will have to pay attention to the colours and geometry of the area in order to recognize new images.

4 Description of the applied machine learning algorithm

For the implementation of the deep learning model we used Keras. Keras is a high-level, deep learning API developed by Google for implementing neural networks. It is written in Python and is used to make the implementation of neural networks easy. It is designed to quickly define deep learning models. Using a Keras model, one of the most important choiches is the optimizer.

4.1 Optimization

Optimizers in machine learning are used to tune the parameters of a neural network in order to minimize the cost function. The choice of the optimizer is, therefore, an important aspect that can make the difference between a good training and bad training [4].

Keras offers different optimizers:

- SGD
- RMSprop
- Adam
- AdamW
- Adadelta
- Adagrad
- Adamax
- Adafactor

- Nadam
- Ftrl

In this section four of them will be analyzed.

4.1.1 Adagrad

It adapts the learning rate to the parameters performing small updates for frequently occurring features and large updates for the rarest ones. In this way, the network is able to capture information belonging to features that are not frequent, putting them in evidence and giving them the right weight. The problem of Adagrad is that it adjusts the learning rate for each parameter according to all the past gradients. So, the possibility of having a very small learning rate after a high number of steps — resulting from the accumulation of all the past gradients — is relevant. If the learning rate is too much small, we simply can't update weights and the consequence is that the network doesn't learn anymore.

4.1.2 Adadelta

It improves the previous algorithm by introducing a history window which sets a fixed number of past gradients to take in consideration during the training. In this way, we don't have the problem of the vanishing learning rate.

4.1.3 SGD - Stochastic Gradient Descent

It is an improved version of batch gradient descent (a more basic algorithm that computes the gradients of the objective function J with respect to the parameters Θ for the entire training set). Instead of computing the gradients over the entire dataset, it performs a parameter update for each example in the dataset. The formula to update weights in SGD is the following:

$$\Theta = \Theta - \eta \cdot \nabla_{\Theta} J(\Theta; x^i; y^i)$$

The problem of SGD is that the updates are frequent and with a high variance, so the objective function heavily fluctuates during training.

4.1.4 Adam

Adam optimizer[6] adds to the advantages of Adadelta and RMSprop (which is very similar to Adadelta, with a difference in the way it manages the past gradients), the storing of an exponentially decaying average of past gradients similar to momentum.

Focus on Momentum

Momentum takes past gradients into account to smooth out the steps of gradient descent. It can be applied with batch gradient descent, mini-batch gradient descent or stochastic gradient descent.

Some advantages of Adam include:

- relatively low memory requirements (though higher than gradient descent and gradient descent with momentum);
- usually works well even with little tuning of hyperparameters.

Conclusion

Optimizers can be divided in two families:

- gradient descent optimizers - *they forces you to manually tune the learning rate*
- adaptive optimizers - *the learning rate is automatically adapted in adaptive algorithms*

Among the four optimizers considered, only the SGD optimizer belong to the family of the gradient descent ones; the other three are adaptive optimizers.

Adam optimizer is the best between those which use adaptive algorithm, that is because it is good with sparse data: the adaptive learning rate is perfect for this type of datasets. Moreover there is no need to focus on the learning rate value. For these reasons Adam optimizer is the best choice in general.

4.2 Inception v3

Inception v3[7] is an image recognition model that has been shown to be over 78.1% accurate in the ImageNet dataset. The model is the culmination of many ideas developed by multiple researchers over the years. The model consists of symmetrical and asymmetrical building blocks, including convolutions, average pool, maximum

pool, concatenations, breaks, and fully connected layers. Batch normalization is used extensively throughout the model and applied to the trigger inputs. The loss is calculated with Softmax.

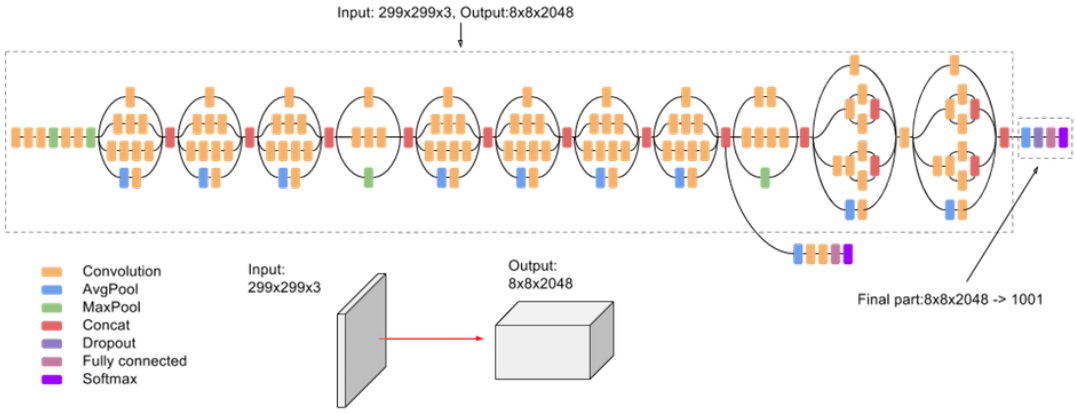

Figure 3: InceptionV3 model[8]

Figure 3 represents the general diagram of the model.

5 Results

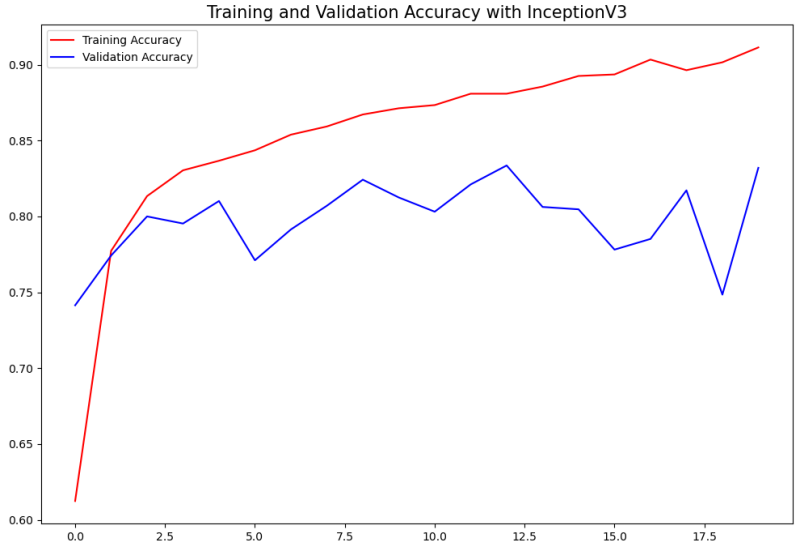

Figure 4: Training and validation accuracy

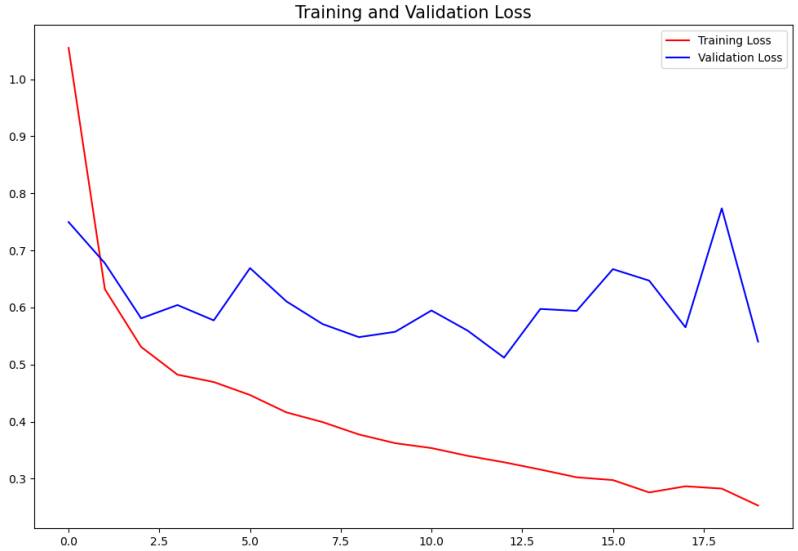

Figure 5: Training and validation loss

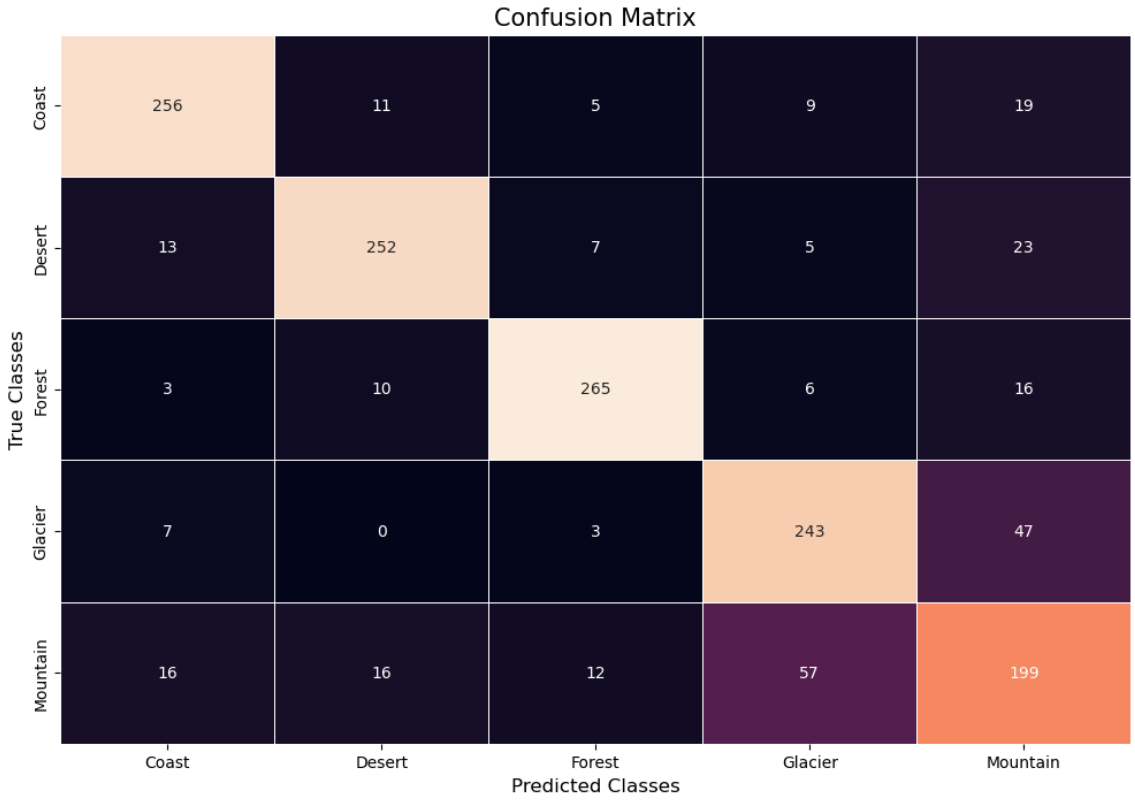

Figure 6: Confusion matrix

The confusion matrix of this model shows that the results are quite good. Most of the errors can be explained: sometimes the model is wrong but, looking more carefully, they are not real errors: for example, last column shows how sometimes the model detect a mountain even if the image represents a glacier or a forest; it is an acceptable error because in the image of a glacier it is possible to see a mountain, and the same happens to the images representing a forest.

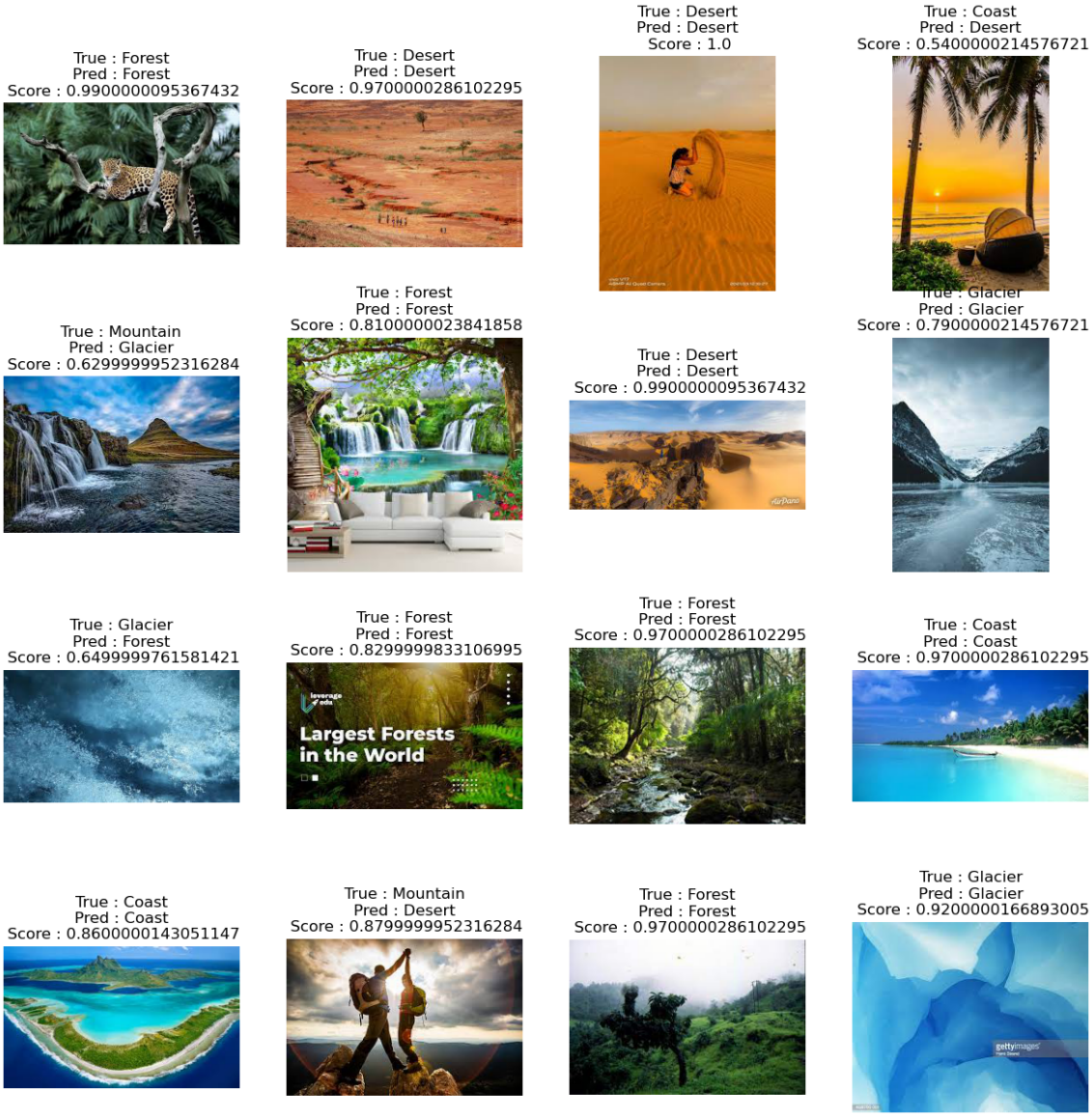

Figure 7: Results

Figure 7 represents a set of results obtained using the pictures from Test Dataset.

5.1 Other methods

The same problem can be addressed using different approaches. The first example[9] is about a solution implemented using $BiT - MR50x1$[10]architecture. This is a model presented in the family of Big Transfer[11], and it is trained to achieve excellent performance on the multi-label classification. The results obtained with this model are axcellent: the training accuracy is 91% with a training loss of 1.4, and the validation accuracy is 84% with a training loss of 3.1.

The second example[12] of a solution of the same problem uses PyTorch. PyTorch is an open source machine learning framework based on the Torch library. Torch is an open source machine learning library used for creating deep neural networks and is written in the Lua scripting language. Using the Adam optimizer, the results are the following: training accuracy of 87% with a training loss of 0.37, and validation accuracy equal to 88% with a training loss of 0.34.

6 Conclusions

The classification of landscapes using deep learning was successfully completed. The reaserch for looking good process was done, found some sources with solving of problem. Model to landscape regognation was trained and it got 0.91 trained accuracy and 0.81 validation f-score accuracy.

The model is not ideal, some of the pictures are predicted as other than from their category. But usually is it due to similarity to another category, less significant for that picture, and for a human it is not a bad prediction.

References

[1] "Landscape recognition | image dataset | 12k images." `https://www.kaggle.com/datasets/utkarshsaxenadn/landscape-recognition-image-dataset-12k-images`.

[2] M. Pak and S. Kim, "A review of deep learning in image recognition," in *2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT)*, pp. 1–3, 2017.

[3] S. K. B. Kieffer, M. Babaie and H. R. Tizhoosh, "Convolutional neural networks for histopathology image classification: Training vs. using pre-trained networks," in *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pp. 1–6, 2017.

[4] H. Lee and J. Song, "Introduction to convolutional neural network using keras; an understanding from a statistician," in *Communications for Statistical Applications and Methods*, vol. 26, pp. 591–610, 2019.

[5] D. Buscombe and A. C. Ritchie, "Landscape classification with deep neural networks," *Geosciences*, vol. 8, no. 7, 2018.

[6] J. B. Diederik P. Kingma, "Adam: A method for stochastic optimization," in *2015 3rd International Conference for Learning Representations*, 2015.

[7] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015.

[8] "Advanced guide to inception v3." `https://cloud.google.com/tpu/docs/inception-v3-advanced?hl=en`.

[9] "Landscape recognition | big transfer | acc : 91%." `https://www.kaggle.com/code/utkarshsaxenadn/landscape-recognition-big-transfer-acc-91`.

[10] "bit/m-r50x1." `https://tfhub.dev/google/bit/m-r50x1/1`.

[11] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby, "Large scale learning of general visual representations for transfer," *CoRR*, vol. abs/1912.11370, 2019.

[12] "Landscape recognition pytorch transfer_learning." `https://www.kaggle.com/code/jiaowoguanren/landscape-recognition-pytorch-transfer-learning`.