Deep Learning Project

MASTER DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS

Final Project Report

Group 16

Beatriz Gonçalves, number: 20210695
Carolina Sá, number: 20210693
Diogo Nobre, number: 20210677
Gonçalo Lopes, number: 20210679
Mariana Romão, number: 20210688

April 2022

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

1. Introduction

The image classification problem, which is concerned with determining the presence of visual structures in an input image, is one of the most important problems in computer vision. Image classification is the process of categorizing data based on the numerical properties of various image features. Many advanced classification approaches, such as artificial neural networks, fuzzy-sets, and expert systems, have been widely used for image classification in recent years, but each has some drawbacks and their accuracy level is comparatively low.

Deep convolutional neural network (CNN) architecture of deep learning is one of the advanced classification approaches that achieves successful results in solving many machine learning problems.

The project's main idea is to apply some of the topics covered in the course to a problem that interests us. CNN was discussed in the course and it became an interesting topic to develop and to learn. With that said, the group selected a classification problem to solve and it will be discussed the task definition, evaluation measures, approach and error analysis.

2. Task Definition

Nowadays image classification is everywhere, if we pay enough attention to our quotidian we can see that it is progressively getting a more important role in our daily tasks. For example searching mechanisms by image are used in many sites and this is one of many possible ways of using this tool. Taking this into account, it has become indispensable to invest and improve in this area that has a lot of room to grow.

Image classification is a complex procedure which relies on different components[1], and so it has many different uses. One of them is the ability to identify areas by their different type of land use (land use being the data that is used for urban planning). This type of data has a very important role in satellite imagery. This type of classification can help scientists understand the impact of changes in the environment of the earth over time, and so it was thought interesting to study it and develop our project around this main issue.

The proposed system takes, as input, an image of the following six classes, as can be seen in Figure 1: buildings, forest, glacier, mountain, sea and street, with the intent of predicting, as output, to each class it belongs. The intention is to reach the highest possible accuracy and to do that several models were tested with several diferenciations on their specifications.

[1] Advances in Domain Adaptation Theory, 2019 (https://www.sciencedirect.com/science/article/pii/B9781785482366500027)

Figure 1 – Examples of images of each class

This system could solve real world problems such as spatial exploration, allowing the possibility to identify the different possible biomes on Earth through satellites or even utilize robots to recognize the existing biomes of Earth in other planets.

Moreover, with the vast number of possible implementations that could be developed, one example would be while driving, where the car would be able to identify the biome and adapt instantly its controls concerning the type of identification.

Another possible implementation would be in aviation where it could be possible to identify the traveled biome and adapt automatically the flying sistem based on that identification, for example, if it recognizes a mountain it would increase the altitude automatically to prevent accidents.

3. Evaluation measure

The search for the dataset that would be selected for this project was not hard, due to the fact that Kaggle has a vastus amount of datasets and so it facilitated our pick. Since it was extracted from that site, it had already been preprocessed and solutions had already been accomplished, making our understanding of the dataset a much more easing step.

Taking a look into the solutions, it was mostly considered accuracy as the main evaluation metric and the results were rounding between 0.85 and 0.95 on accuracy, however, the solutions also included the loss metric, the running time and other important metrics to consider.

For the project, the group took running time into consideration, because of the various parameters and arquitectures. Although, the problem at hand does not require especially fast algorithms since it's not done in real time.

Our final model obtained an accuracy of 0.8587, a loss of 0.4601, taking around 196 seconds per epoch for validation. Unfortunately, in comparison to the other Kaggle solutions, our final model barely accomplishes similar results, since most of them used VGG16, a Convolutional Neural Network architecture we decided against mainly due to the long running time.

4. Approach

Our dataset is composed of approximately 25 thousand images: 14 thousand for training, 3 thousand for validation and 7 thousand for testing. Based on the size of the dataset, the network will hopefully generalize well to new images and classify them into correct categories. However, the dataset can also impose limitations, such as running time.

Our project is a computer vision problem and so, the appropriate deep learning model for the task is a CNN (Convolutional Neural Network). With that in mind, a simple model was created, at which various alterations were tested, such as altering:

- The number of convolutional layers;
- The number of filters as well as their size;
- The padding;
- Adding dropout layers;
- The number of dense layers and the number of neurons in each;
- The optimizer;
- The number of epochs;
- The steps per epoch;

All these alterations, along with others, had in account the running time, the accuracy and loss in training and validation to create a balanced and efficient model.

The first step for our project was to prepare the data. The data preprocessing process consists of loading batches of 32 images from their directory into the specified set, training and validation, decoding the images to RGB grid of pixels, resizing them to 150x150, rescale the pixel values, between 0 and 255, to values between 0 and 1 and assigning the correct class to each image. With data augmentation, each image is also slightly altered at random based on various parameters.

Initially, data augmentation wasn't used just to verify the results, and as expected, the model overfitted, even with the use of dropout layers, however it got significantly good results with just a few epochs so it could be a good solution for our problem. Even so, data augmentation is good practice and after it was applied there wasn't overfitting even without the use of dropout layers.

Besides creating a new model, the implementation of the VGG16 architecture was also tested, nevertheless the running time required for each epoch made it unviable.

Since the dataset is composed of 6 different classes, for any model tested, the last dense layer consisted of 6 neurons with the softmax activation function and the loss was calculated based on categorical cross entropy.

In conclusion, the model implementation had various limitations because of the type and size of the dataset, it being a computer vision dataset with 6 different classes. These limitations can be overcome by utilizing GPU and other tools to fasten the running time, making it possible to obtain much better results and even try new models, keeping the same or even faster running times.

```
model_1 = Sequential()
model_1.add(layers.Conv2D(16, 3, padding='same', activation='relu', input_shape=(150,150,3)))
model_1.add(layers.MaxPool2D())
model_1.add(layers.Conv2D(32, 3, padding='same', activation='relu'))
model_1.add(layers.MaxPool2D())
model_1.add(layers.Conv2D(64, 3, padding='same', activation='relu'))
model_1.add(layers.MaxPool2D())
model_1.add(layers.Flatten())
model_1.add(layers.Dense(128, activation='relu'))
model_1.add(layers.Dense(6, activation='softmax'))

model_1.compile(loss="categorical_crossentropy",
              optimizer=Adam(),
              metrics=['accuracy'])

model_1.summary()
```

Figure 2 – Layers of the model

```
history_1 = model_1.fit(train_data,
                        epochs=10,
                        batch_size=32,
                        steps_per_epoch=len(train_data),
                        validation_data=test_data,
                        validation_steps=len(test_data)
                        )
```

Figure 3 – Feting parameters of the model

5. Error analysis

As a matter of fact, the Validation correctly predicts about 85% of the time. As can be seen in Figure 4, it is important to note that sometimes streets are confused with buildings, due to the fact that buildings also appear in the image.

In addition, the Test does not show true classification. However, by observing Figure 5, it is possible to analyse the results and find some errors. Namely, in the first image, a mountain is identified instead of sea, and, in the third image of the first row, we can observe a sea bottom, but a glacier is identified.

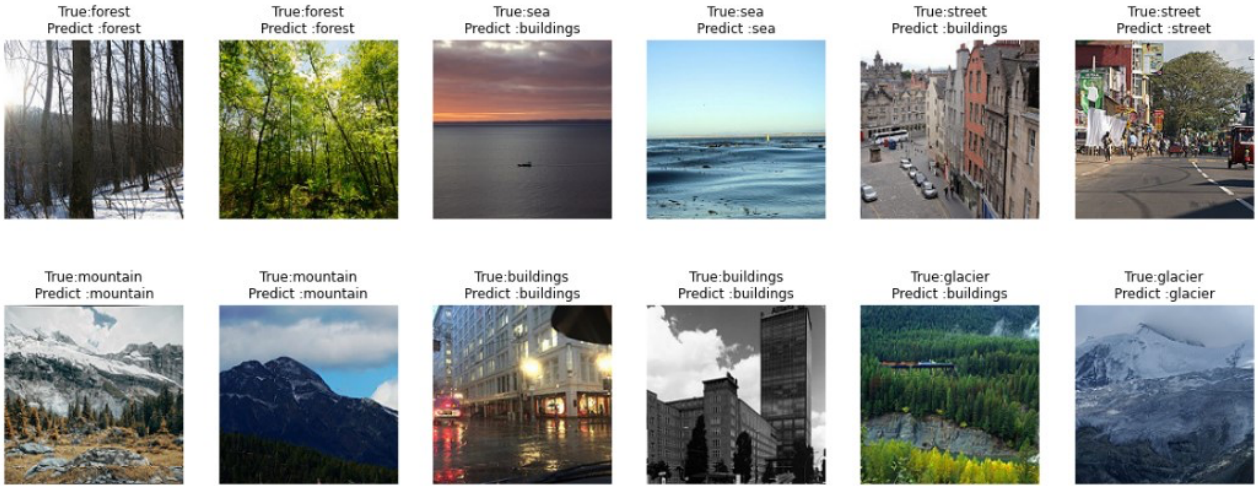

Figure 4 – Output of Validation predictions

Figure 5 – Output of Test predictions

As can be seen in Figure 6, the validation loss reaches its minimum, 0.41, in the last epoch, while the training loss continues to decrease until it reaches approximately 0.46. In turn, Figure 7 shows that training accuracy increases linearly over time, until it reaches about 83%, while validation accuracy increases until 86%.

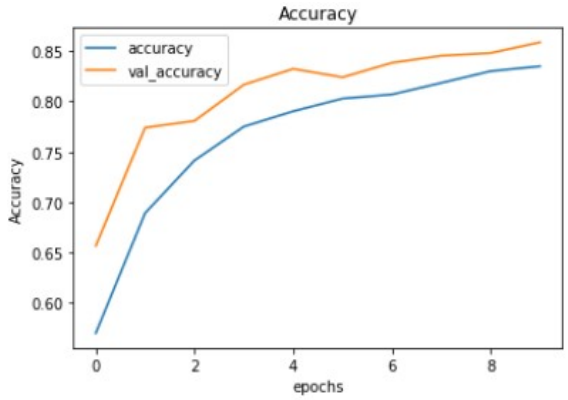

Figure 7 – Accuracy plot

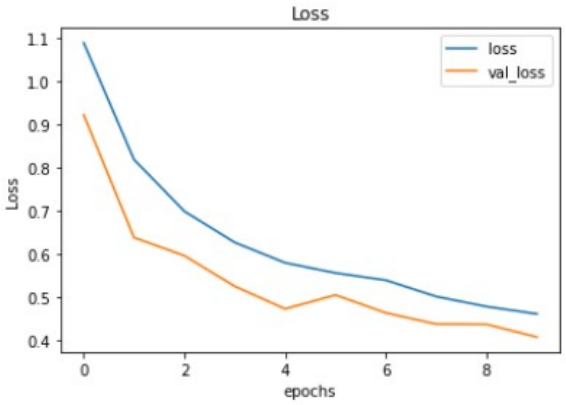

Figure 6 – Loss plot

In fact, our model has an accuracy of approximately 83% on the training set and about 86% on the validation set. This means that the model is expected to have an accuracy of approximately 86% on the new data. It should be noted that as the epochs go from 0 to 9, the accuracy metric increases, while the validation accuracy metric also increases at the approximately the same rate. This means there is no overfitting, being a result of using the Image Data Generator function, without it, we need to use dropout layers to combat overfitting in the model.

In addition, the take-away message is that our model has good predictive capabilities on both the Validation and Test levels even though it has some flaws, as predicted by the model's accuracy. Moreover, metrics such as precision and recall were also calculated with the purpose of getting a more insightful knowledge of the predictions of each class, as shown in Figure 8.

Classification Report

	precision	recall	f1-score	support
buildings	0.85	0.84	0.85	437
forest	0.96	0.97	0.96	474
glacier	0.83	0.81	0.82	553
mountain	0.81	0.80	0.81	525
sea	0.86	0.86	0.86	510
street	0.85	0.89	0.87	501
accuracy			0.86	3000
macro avg	0.86	0.86	0.86	3000
weighted avg	0.86	0.86	0.86	3000

Figure 8 – Classification Report

Concluding, the model seems to fail more, as already mentioned, in the distinction between buildings and streets, since it is common for a street to contain buildings around it, also glaciers and mountains, because of their visual similarity. Furthermore, the approach seems to predict less the sea biome, as it is possible to see in Figure 9.

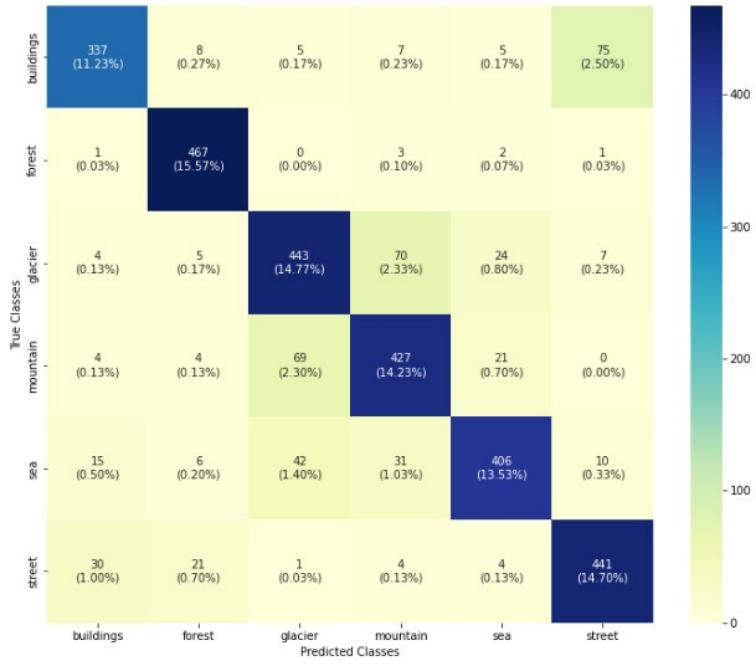

Figure 9 – Confusion Matrix

6. Conclusion

Image classification in machine learning is used to predict the class membership of the unknown data instance based on the class membership of the training data, which is known[2]. It is important to note that one of the algorithms used was CNN.

Convolutional Neural Networks is an advanced classification algorithm that with a simple model can achieve really good results for computer vision problems, like our task for this project, that is centered in the application of a CNN model in classifying six types of classes represented in images.

In a perfect world we could have developed a perfect model. As it is not possible, it is known that the model can be improved by adding new data to the set and expanding the classification options. In the future, with the rapid development of new technologies it will become possible to develop more complex models with the ability to overcome many adversities and limitations that are faced nowadays.

7. References

1. Redko, I., Habrard, A., Morvant, E., Sebban, M. and Bennani, Y. (2019). Domain Adaptation Problem. Advances in Domain Adaption Theory, [online] pp.21–36. Available at: https://www.sciencedirect.com/topics/engineering/image-classification
2. Sinha, A. (2016). Hadoop in the Cloud to Analyze Climate Datasets. Cloud Computing in Ocean and Atmospheric Sciences, [online] pp.245–276. Available at: https://www.sciencedirect.com/science/article/pii/B978012803192600013X
3. Gavali, P. and Banu, J.S. (2019). Deep Convolutional Neural Network for Image Classification on CUDA Platform. Deep Learning and Parallel Computing Environment for Bioengineering Systems, [online] pp.99–122. Available at: https://www.sciencedirect.com/science/article/pii/B9780128167182000130
4. Thakur, R. (2019). Step by step VGG16 implementation in Keras for beginners. [online] Medium. Available at: https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c
5. Bansal, P. (2019). Intel Image Classification. [online] Kaggle.com. Available at: https://www.kaggle.com/datasets/puneet6060/intel-image-classification
6. niladri54 (2021). Intel-Image-Classification using cnn. [online] Kaggle.com. Available at: https://www.kaggle.com/code/niladri54/intel-image-classification-using-cnn

[2] Advances in Domain Adaptation Theory, 2019 (https://www.sciencedirect.com/science/article/pii/B9781785482366500027)