SDS 2020 - Module 3: Group Assignment
Group members: Lukas Bielmeier, Marta Fedeli

STAKEHOLDER REPORT

Multiclass Image Classification using Keras Deep Learning

1. Problem formulation

One of the biggest challenges with regards to sustainable development is for an economy to grow without exhausting the present natural resources. Monitoring the development patterns of land use and land cover is one of the ways to explore whether this delicate balance is being observed and maintained. Land use (and its change) can in fact have severe environmental impacts on ecosystems, as well as on food provision, the nutrient cycling and climate change mitigation because of reduced carbon sinks (forests). In this case, monitoring through satellite land cover data can (and must) drive action, for better informed land use policies and by ensuring that our shared natural capital is well managed, it can help maintain a healthy environment and human well-being (EEA, 2020).

As students of Development & IR and Innovation & Sustainable Development, we are naturally drawn to a work that provides an impact, either socially or with regard to the environment. Within the scope of this assignment, we aimed at designing a deep learning model for accurately predicting the content within a picture - in this case, to differentiate between what is forest land, what appears to be an industrial area, and so on.

We believe that doing so could provide crucial insights for urban planning or help in creating a circular economy that can draw from its neighboring natural resources without exhausting them.

Having a model that can successfully classify whether or not a particular satellite image is depicting a forest or industry, would mean monitoring will not need to be handled manually anymore. Not only, but potentially, with enough training data, the model could become more accurate in its manual classification.

This kind of model might be especially useful when combined with time series data. By analysing a predefined region through time, a deep learning model that can automatically categorize satellite images could detect development patterns that potentially bear interesting social and environmental insights, as well as with regards to past development, showing for example whether or not the pace of deforestation has increased or decreased. As the sustainable use of natural resources and the transition to a circular economy become more popular topics in academia and industry alike, we think that a model like ours could potentially add a great deal of value to people analysing such development patterns. Our main motivation for this analysis is hence to contribute to this quickly developing field and to try to bridge the gap between development studies and data science methods.

2. Choice of data

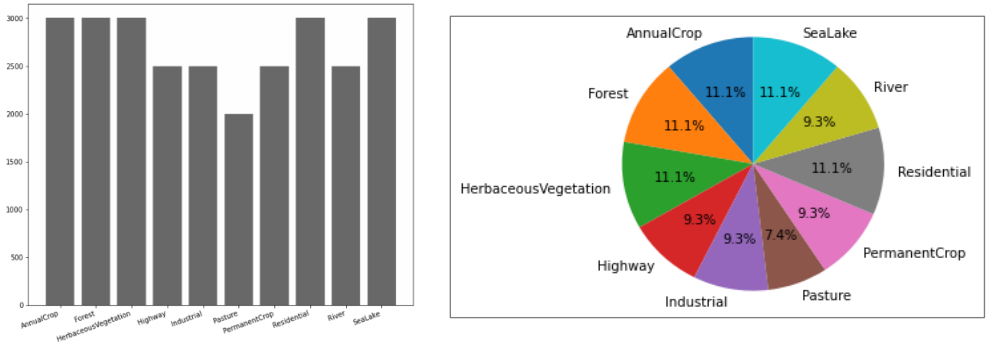
The dataset at hand was accessed and loaded from Patrick Helber's "EuroSAT" repository on Github. The EuroSAT dataset consists of 10 classes with 27000 labeled and geo-referenced samples of satellite images. Although the dataset came already split into ten labelled classes, image data is not so easily accessible as for example tabular data. Therefore, some additional pre-processing steps were required. Specifically, in our case, it needed to be streamed directly from its directory path and required splitting the ten folders into *train and test* subfolders, as well as the image augmentation process performed with *ImageDataGenerator*, which are steps needed for working with the classification model later.

3. Exploratory analysis

We first performed a brief exploratory analysis, to gain insights into our data as well as to investigate what type of images and distribution occurs among classes.

Image per class – examples

It is good practice, especially when tackling a classification problem, to make sure one's dealing with a well balanced dataset: were this not the case, we could incur in problems of false accuracy.



Sample distribution per folder (absolute and relative)

As shown above, we plotted a bar plot and a pie chart that proved, by showcasing the volume of each folder, as well as their distribution across, that the dataset was well balanced and fit for our model.

4. Non-neural baseline model

4.1. Feature extraction and image pre-processing

Neural networks and the deep learning models are rather state-of-the-art when it comes to image classification models and especially Convolutional Neural Networks (further defined down below) have the highest performance when tackling this task. To prove how much more beneficial and efficient building a deep learning model is in comparison to a baseline non-neural one, we trained a RandomForestClassifier on our dataset and analyzed the results.

However, before doing so, pictures need to be pre-processed and we did so with the help of *scikit-image.* This collection of algorithms works similarly to dimensionality reduction, compressing the high number of information stored in a picture's pixels into few significant features. Whether is by edge detection (which surfaces object's structures in a picture) or simply through reading the pixel's raw values, *scikit-image* processing converts a visual representation to a numerical one, returning an array, which is exactly the format the data needs to be in to be run through a RandomForestClassifier.

4.2. RandomForestClassifier

Once ran, the RandomForestClassifier achieved a rather poor performance, with an accuracy score of a little over 50%. Moreover, as it is shown in the notebook's confusion matrix and classification report, precision and recall scores strongly differed across different classes. This could be due to the fact that while some classes contain image features that particularly stand out to the model, others show more ambiguous ones. We can therefore conclude that the model was not up to the task and can move onto building a state-of-the art deep learning model.

5. Deep learning CNN model

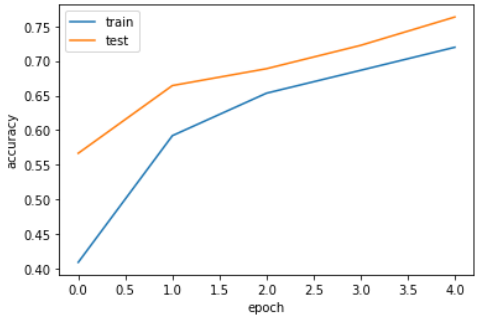
Finally, to tackle the core goal of our assignment, we chose to build a Convolutional Neural Network (CNN) as our multi-class image classification model.

Like above mentioned, working with images first requires some sort of dimensionality reduction and CNNs are very effective in reducing the number of parameters automatically, without losing on the quality of the models. Already, this represents quite the improvement in comparison to non-neural baseline models, especially in terms of preprocessing efforts.

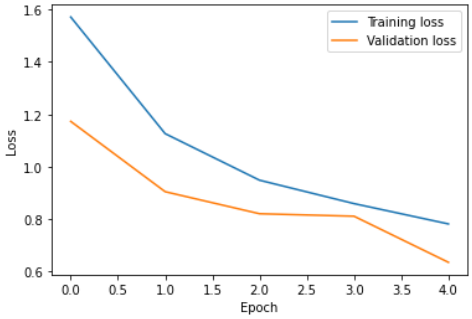
We started designing our model with a convolutional layer to filter our image input and create a map summarizing feature presence in the given input. However, feature maps are sensitive to where the features are located input-wise and need to be therefore sampled down through a pooling layer. For this reason, all of the three convolutional layers we added in our model are respectively followed by *MaxPooling2D* layers.

Then, a flatten layer was added to collapse the spatial dimensions of the input into the channel dimension, followed by two dense layers, there to feed all outputs from the previous layer to all their neurons, each neuron providing one output to the next layer.

Once the model was compiled, we fitted it to the *training_set*, achieving a model accuracy of 75% (almost 20% higher than the RandomForestClassifier).
Interestingly, the validation accuracy appeared to score higher than the training one. This is due to the fact that, as we have previously augmented the training set to a larger extent in comparison to the test set (where images were only rescaled), our models feeds into a more varied training set, making it harder to accurately classify a certain picture in the training set than in the test set.
To ensure that the model is not overfitted, we then proceed with plotting a graph of the model accuracy and loss as it was training.



CNN model accuracy

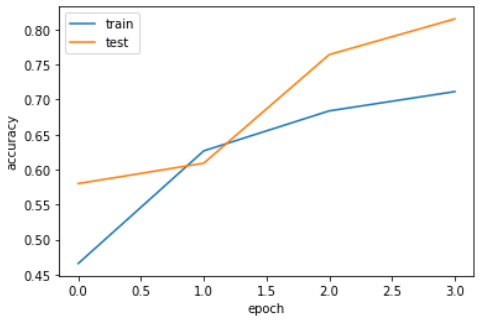


CNN model loss

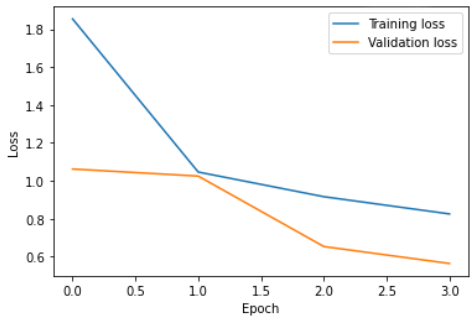
This appears not to be the case, as the above graphs show a level of accuracy steadily rising for both the train and the test set, while the loss (difference between the model predicted value and the true value) seems to fall for both sets over epochs. Another sign of "good health" of our model is the slopes' behavior in both graphs: steeper at first and getting flatter with each epoch, as well as moving closer together.
In conclusion the model proved to be fairly successful; however, in the next section we will present how tuning further improved the model performance and raised its levels of accuracy.

5. Transfer Learning tuning

To enhance the model ability to classify and assign satellite images to their correct classes, we proceeded with tuning our model through Transfer Learning (TL). TL optimizes the learning portion of a new task by transferring previous knowledge from a related task that has already been learned, allowing for rapid progress and improved performance of the model. To do so, we loaded the premade CNN model VVG16 and added two new layers to the previously existing model: only these two will be trainable, thus ensuring that most of the information already embedded in the original model is saved.
Immediately, when looking at the TL model's accuracy and loss values, there is improvement on both sides: as the loss fell from 0.64 to 0.57, the accuracy score rose from 0.76 to an accuracy of over 0.80. The accuracy and loss graphs plotted for this model and displayed below, prove that, as accuracy is constantly rising and loss is constantly falling throughout the epochs, TL proved to be a successful tuning which bettered the classification power of our CNN model.



TL model accuracy



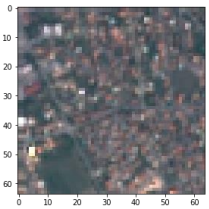
TL model loss

6. Model prediction tests

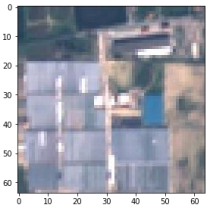
Finally, we proceed to test our model classification power with random pictures chosen out of the satellite images at hand. Out of the four tests performed, two were carried out with the original model and two with TL tuned model.

Out of all the four tests, only one was predicted accurately (TL model test). As above mentioned, the model accuracy was raised from one of 76% to one of 80% and although it could have been perhaps further raised by additional tuning, it was still a fairly good performance, hence why the tests results are somewhat disappointing.

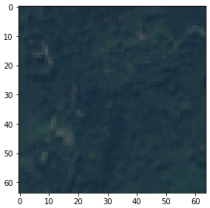
However, when actually looking at the test pictures (which are displayed here below), there seems to be somewhat of an explanation to why these tests failed which ties into the topic of data's quality.



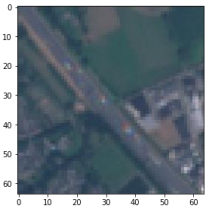
Residential (1st example)



Industrial (2nd)



Forest (3rd)



Highway (4th)

First of all, the picture sample lacks high pixels' resolution and the objects or patterns in the images are not particularly well defined. The silver lining is that out of the 10 classes, even when the model predicted the wrong label, it still predicted a class that shared similarities with the picture's true label. For example, by labeling a forest as a pasture the model was still "smart enough" to classify as something that could pass for a forest and not a River or an Industrial setting. Same goes for the mislabeled Residential picture, which is not completely far off from an industrial setting (in comparison to other classes). Lasty, when looking at the Highway picture, we can actually see how the road's surroundings highly resemble pasture lands.

In conclusion, since the model achieved a fairly good level of accuracy, both before and even more so after tuning, it could be argued that if in the future, higher quality satellite pictures were to be used, it would help achieve a more accurate classification for land use and land cover.