



DEPARTMENT OF MECHANICAL ENGINEERING

INTELLIGENT SYSTEMS - SINT

Convolutional Neural Network as a Method for Identifying Brick Kilns in Bangladesh

Authors:

Thomas Lunde Fosen (ist1101349)
Anders Sørskår Larsen (ist1100887)

November, 2021

Table of Contents

1	Problem Statement	1
2	Related Work	2
3	Data	3
3.1	Overview	3
3.2	Pre-Processing	3
4	Model	5
4.1	Overview	5
5	Results and Discussion	8
5.1	Results and Evaluations	8
5.2	Discussion	10
6	Further Work and Improvements	11
7	Conclusion	12
	Bibliography	13

1 Problem Statement

As a part of their stride to increase global prosperity, the United Nations have developed 17 Sustainable Development Goals (SDG), in order to concretize their efforts. The work and study conducted for this assignment is an implementation of a possible solution for one of the aforementioned goals, more precisely, SDG13: Climate Action.

One of the major obstacles obstructing the progress of the SDGs historically has been a lack of key data and information, especially in less developed parts of the globe. Modern development and innovation within fields such as machine learning and intelligent system have provided a new angle of approach toward creating viable solutions, by using more abundant and accessible data, such as GPS imagery, and social media data.

As a way of facilitating and monitoring advancements within these fields, a collection of 15 benchmark tasks across 7 SDGs have been introduced, through the 'SustainBench' project. The motivation for introducing the project was based on three underlying goals, namely to;

1. lower the barriers to entry for the machine learning community to contribute to measuring and achieving the SDGs
2. provide standard benchmarks for evaluating machine learning models on tasks across a variety of SDGs
3. encourage the development of novel machine learning methods where improved model performance facilitates progress towards the SDGs

The following report is a study and overview of our attempt to develop a viable solution to one of the aforementioned benchmark tasks, namely task 13A: Brick Kiln Classification. A necessity for combating climate change is a strong informational foundation regarding real-time emissions and the state of pollution worldwide. Certain high-carbon industries, particularly in less developed parts of the world, are not sufficiently monitored due to their informal and unregulated nature, making it difficult to adequately handle their environmental impact.



Figure 1: Brick kiln as seen from a satellite image.

A recent novel approach to tackle this issue, has been the use of satellite imagery, combined with machine learning methods as a way of identifying and locating such manufacturing plants. Our implementation of a neural network will be designed to identify brick kilns across Bangladesh using the aforementioned imagery.

2 Related Work

This segment is intended to provide a reference as to the current state of research withing this specific application, and hopefully to help clarify our works relevance in relation to the existing field and publications.

The recent developments within computing power and complexity have contributed to innovations and strides within the machine learning field, with neural networks becoming more and more viable and legitimate method for performing tasks involving large amounts of data. Several attempts at providing high-performing neural networks for application withing brick kiln identification in South Asia have been conducted and publicized by different academical entities and researches over the last few years, with further innovations expected in the future.

Recently, researchers from Stanford University published a report in collaboration with scientists from Bangladesh, called "Scalable deep learning to identify brick kilns and aid regulatory capacity". [Jihyeon Lee, et al. 2021]. In order to help improve compliance with environmental regulations, they set out to develop a scalable deep learning method for identifying brick kilns across South Asia. In addition to classifying images containing kilns, they also developed a method for locating the detected kilns geographically. The visual result is shown in figure 2, where the red structures in the photo are identified brick kilns in the Khulna District in Bangladesh, with their coordinates plotted as green dots.

They were able to produce a method which proved highly beneficial as compared to traditional methods, and the results they obtained helped determine that machine learning approaches are a valuable resource. Their developed model was deemed by the team to be "quick, inexpensive, reproducible, and easily scalable", and, in doing so, contributed to establishing groundwork for this particular application, and proving the method viable.

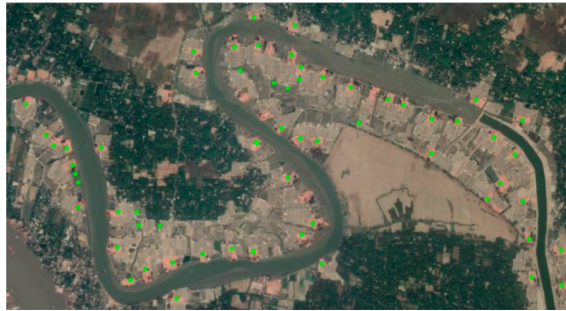


Figure 2: Identified brick kilns.

Another group of researchers, who claim to have outperformed other state-of-the art methods, published their findings in a report called "Kiln-Net: A Gated Neural Network for Detection of Brick Kilns in South Asia". [Usman Nazir, et al. 2021]. They developed a gated neural network consisting of a two networks, where the first filters out 99% of the images not containing kilns, and the other locates the classified kilns geographically, namely a classifier and a detector.

Their final solution obtained an accuracy score of 99.96%, and a F1 score of 0,91. As well as obtaining these results, the team deemed their method to be roughly 20x faster than previous methods published, meaning it was a vast improvement in regards to computational cost. These findings once again helped confirm the legitimacy of machine learning methods within such applications, and emphasized the importance and likelihood of innovations to the field.

The scale and scope of our report and study is far less comprehensive than those of the previously discussed reports, due to complexity and workload limitations. Thus, our work will focus on the image classification aspect, by developing a model for recognizing whether or not an image contains a brick kiln. Another obvious advantage as to our study, is the fact that an already labeled and classified, processed, data-set was provided to us through the formerly introduced "SustainBench" project.

3 Data

3.1 Overview

Due to storage limitations, the online notebook environment Kaggle was used, for running and training the model, utilizing the CPU and GPU capacities available. However, due to spacial and computational limitations within this environment, we were only able to use a subset of the data-set, - 40% of its original size. We found that this was not a limitation with regards to the computed results.

The public data-set provided for this task consists of low-resolution images from the Sentinel 2 satellite constellation from the Google Earth Engine, taken between October 2018 and May 2019. Roughly 73 thousand images are provided in the set, with 6,329 containing a brick kiln, and 67,284 not containing any kiln. The 64x64 (10m resolution) images are stored as HDF5 files, alongside various metadata relating to the image. Originally, the data contains information regarding the geographical location of each image, which can be used to locate potentially identified brick kilns. This task is considered out of scope of this paper, and will thus not be implemented, although it is worth mentioning that this is a possible extension for future work.

The HDF5 files each contain the following four keys;

1. Bounds: Longitude and latitude of top-left and bottom-right corner.
2. Images: Image Data
3. Indices: Global position of image
4. Labels: Classification value (1 if image contains kiln, 0 otherwise)

3.2 Pre-Processing

Sustainbench has published a GitHub repository [*SustainBench GitHub* n.d.], which provides data-loaders and baseline models for most of the benchmark tasks, as well as processed versions of the necessary datasets. We used the supplied functionality for several aspects of the data processing, which was helpful in getting the data to the desired format.

13 bands are retrieved from the Google Earth Engine Catalog, where bands 2, 3 and 4 are extracted to obtain the desired RGB values of the image. Initially we had to use the function `get input` and pick the bands which represented RGB. After doing so, we end up with images with shape: (num of instances, img height, img width, color channels)=(,64,64,3) for each split of the data-set.

In addition, the input data was normalized, in order for the CNN to work on smaller values in the range [0, 1]. This is done using the following equation:

$$Normalized = \frac{value - minValue}{maxValue - minValue} \quad (1)$$

In the above equation, "value", "minValue" and "maxValue" refers to the pixel value at every location in the 64x64 image, the minimum pixel value and the maximum pixel value respectively.

The task at hand is a binary classification task, with class 0 - no brick kiln, and class 1 - brick kiln. The data is provided as a train, validation and test split of 80-10-10 that preserves the relative proportions of classes across the splits. However, the class instances is not shuffled - every instance of class 1 is rather located at the very end of each split. Therefore, we shuffle the instances of each split.

Finally, and most importantly, the issue of skewed class distribution had to be dealt with. Binary classifiers often give misleadingly optimistic performance on classification of an imbalanced class

distribution. Machine learning algorithms such as the one used in this project, a neural network, tend to learn that the minority class is not as important as the majority class. This is problematic because the minority class is the one that we are most interested in classifying - namely the instance in which there exist a brick kiln. One solution to this problem is to introduce class weights. Scaling the weights for both classes helps keep the loss to a similar magnitude, and causes the model to pay more attention to instances from the minority class.

$$Weight(0) = \frac{1}{Negatives} * \frac{Samples}{2} \approx 0.55 \quad (2)$$

$$Weight(1) = \frac{1}{Positives} * \frac{Samples}{2} \approx 5.99 \quad (3)$$

Equations 2 and 3 show the method used for calculating the respective class weights, with their approximated final value. As seen in the values, the samples containing a brick kiln is weighed with a factor of roughly 10, compared to the negative samples.

Processing and treating raw data is one of the major challenges in developing real-life solutions, and is something that needs to be tailored to the specific aims and applications of every solution. In this specific case, obtaining the data in the desired format was rather challenging, although, with the help of the sustainbench public functionality, it was manageable.

4 Model

4.1 Overview

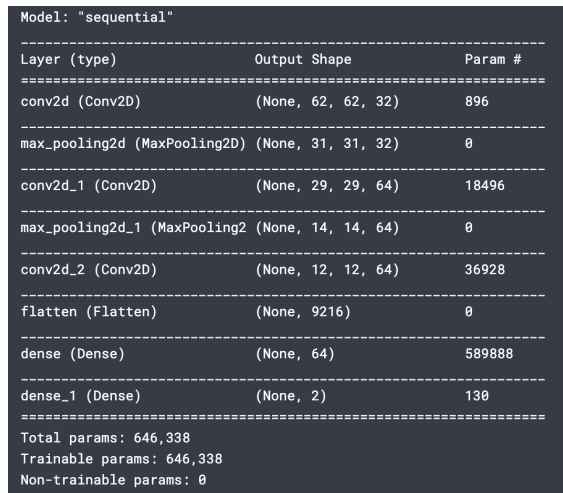
The chosen model for this classification task is a variant of a neural network, specifically a Convolutional Neural Network (CNN). The CNN uses 2D convolution layers in its network to convolve learned features with the given input data, which means that it is ideal for processing 2D images like the ones in this particular data-set.

A general CNN consists of an input layer and an output layer, along with a specified number of hidden layers, where, as aforementioned, the hidden layers consist of convolutional layers. Additionally, pooling layers and fully connected layers are added in order to combine clustered neuron outputs into single neurons in the next layer, and to connect every neuron in one layer to every neuron in the next layer, respectively.

When creating the model, a CNN with a basic architecture was implemented, in order to get an impression of its ability to classify images. The idea was to alter and iteratively improve the architecture and test different structures in order to get the best possible results afterwards. However, in terms of both accuracy, recall and precision, the model yielded surprisingly good results, and was thus deemed satisfactory for modeling and interpreting of results.

As to be discussed regarding further work and improvements, the approach of iteratively improving a working model is good practice, and is something that would be done in this setting application, was it to be further developed. Due to time- and work-constraints, this was not feasible at this point in time.

A convolutional base with a common pattern of Tensorflow's Keras API Conv2D and MaxPooling2D layers was implemented in the model. The CNN takes tensors as input of shape (image height, image width, color channels). These tensor are created in the data pre-processing step, previously described in section 3.2.



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 29, 29, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_2 (Conv2D)	(None, 12, 12, 64)	36928
flatten (Flatten)	(None, 9216)	0
dense (Dense)	(None, 64)	589888
dense_1 (Dense)	(None, 2)	130
Total params: 646,338		
Trainable params: 646,338		
Non-trainable params: 0		

Figure 3: Model Architecture

In order to perform classification two dense layers were added. These layers take 1D vectors as input, and therefore a flatten layer was also implemented, which unrolled the 3D output to 1D. The dense layers were added on top of the flatten layer. Due to the task at hand being a binary classification, the final dense layer was configured with two outputs. The architecture of the model is shown in figure 3.

As opposed to a fully connected network, CNNs and its convolutional layers consists of neurons which has local receptive fields. This means that the neurons are not connected to the entire input, but rather some section of the input. These input neurons provide abstractions of which taken

together over the whole input represent what is called a feature map. Another important feature of convolutional layers is that they are spatially invariant, which means that they look for the same features across the entire image. Neurons in a convolutional layer that cover the entire input and look for these features are called filters.

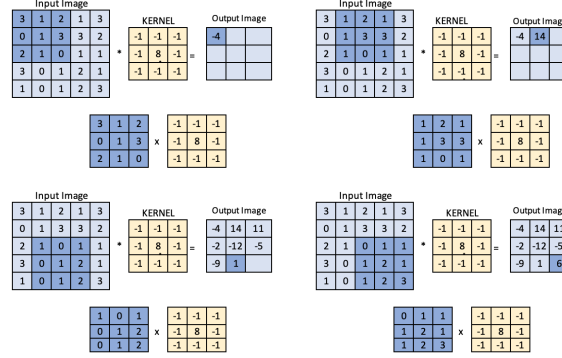


Figure 4: Example of convolution on a 5x5 image, with 3x3 kernel size.

The first convolutional layer, i.e. the input layer, takes in an input of shape (64,64,3) and applies 32 filters each of kernel size 3x3. Keras configures each filter as (3,3,3), i.e. a 3D volume covering the 3x3 pixels plus all the color channels. Each filter also has an additional weight for a bias value. Hence, the filter on a (64,64,3) input produces a (62,62,32) output. This can be thought of as 32 images, one for each filter, each 62x62 pixels.

In a standard convolution process the filters are not applied to the borders. To adjust for this, padding is often used. Also, one could define stride, which is how far the filter moves when computing the next pixel. The default values for these parameters were used in our model, i.e. no padding and (1,1) stride.

ReLU (Rectified Linear Unit) was chosen as the activation function in each convolutional layer and in the dense layer, as this is recognized as a well-performing standard function for classification applications.

The MaxPooling2D layers added between the Conv2D layers reduces the number of parameters and hence reduces the computational complexity of the network. By progressively reducing the spatial size, it ensures that overfitting is somewhat controlled.

Before training, the model must be compiled, and a loss function needs to be defined. Sparse Categorical Crossentropy, which computes the cross-entropy loss between labels and predictions, was utilized in combination with an Adam Optimizer, which is a stochastic gradient descent optimization algorithm for training the model, before fitting the model to the training data, for a predetermined number of epochs and batch size. Then, a probability model is created, where predictions can be computed. The steps are visualized in the below code-snippet.

```
modelCnn = createCnn()
modelCnn.summary()

modelCnn.compile(optimizer='adam',
                  loss=tf.keras.losses.SparseCategoricalCrossentropy
                    (from_logits=True),
                  metrics=['accuracy'])

history = modelCnn.fit(x_train_norm, y_train,
                       epochs=EPOCHS,
                       batch_size=BATCH_SIZE,
                       validation_data=(x_val_norm, y_val),
```

```
class_weight=class_weight)

probability_model = tf.keras.Sequential([modelCnn,
                                         tf.keras.layers.Softmax()])

def createCnn():
    img_height = 64
    img_width = 64
    color_channels = 3

    model = models.Sequential()
    model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(img_height,
                                                                           img_width,
                                                                           color_channels)))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))

    model.add(layers.Flatten())
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dense(2))

    return model
```

5 Results and Discussion

5.1 Results and Evaluations

Six evaluation methods will be used as benchmarks for evaluating the feasibility of our implemented model, specifically overall accuracy, precision, recall, F1 and AUC score. These scores are formulated in terms of true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN), and are visualized in formulas 1 through 4.

Loss is a quantitative measurement of the deviation of the predicted output compared to the actual output. The AUC score represents the area under the ROC curve, which shows the performance of a classification model at all thresholds. The class attribute refers to the classifications, meaning 0 for no kiln, and 1 for kiln.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

One thing to note regarding accuracy, is that this measurement is a rather inefficient measurement for an imbalanced data-set, meaning, when there are far more negative than positive samples in the set. In this scenario, the accuracy score is high for a classifier merely classifying every instance as negative, or mislabeling most, if not all, of the positive samples, as the total number of true negatives outweigh the total number of true positives by a large factor. Therefore, it is important to regard this measurement in combination with other measurements. This fact has also been accounted for, as mentioned in section 3.2.

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

$$Recall = \frac{TP}{TP + FN} \quad (6)$$

$$F1 = \frac{TP}{TP + 1/2(FP + FN)} \quad (7)$$

Listed below are the final results of the model being run on a variation of epochs and batch sizes. The model was run on several distinct epoch and batch size configurations, in order to study and evaluate their effect on the final results. These listed results in the tables below are from running the model on the test-set.

Class	Precision	Recall	F1	AUC	Loss
0	0.98	0.93	0.96	0.89	0.20
1	0.55	0.85	0.67	0.89	0.20

Table 1: Overall accuracy = 0.92, Epochs = 10, Batch Size = 128

Class	Precision	Recall	F1	AUC	Loss
0	0.99	0.89	0.94	0.91	0.29
1	0.44	0.92	0.60	0.91	0.29

Table 2: Overall accuracy = 0.89, Epochs = 10, Batch Size = 256

Class	Precision	Recall	F1	AUC	Loss
0	0.98	0.95	0.97	0.88	0.25
1	0.60	0.84	0.70	0.88	0.25

Table 3: Overall accuracy = 0.94, Epochs = 15, Batch Size = 128

Class	Precision	Recall	F1	AUC	Loss
0	0.98	0.98	0.98	0.90	0.14
1	0.77	0.80	0.78	0.90	0.14

Table 4: Overall accuracy = 0.96, Epochs = 15, Batch Size = 256

Class	Precision	Recall	F1	AUC	Loss
0	0.97	0.96	0.97	0.84	0.23
1	0.66	0.71	0.68	0.84	0.23

Table 5: Overall accuracy = 0.94, Epochs = 20, Batch Size = 128

Class	Precision	Recall	F1	AUC	Loss
0	0.99	0.96	0.98	0.91	0.18
1	0.70	0.86	0.77	0.91	0.18

Table 6: Overall accuracy = 0.95, Epochs = 20, Batch Size = 256

A confusion matrix was also calculated in the case of 15 epochs and batch size of 256, in addition to precision and recall. This is another way of illustrating how the classification model is performing and characteristics of the predictions it is making. This matrix is visualized in figure 5.

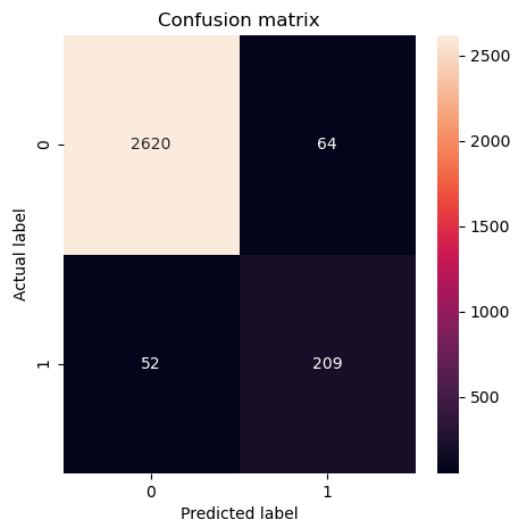


Figure 5: Confusion Matrix

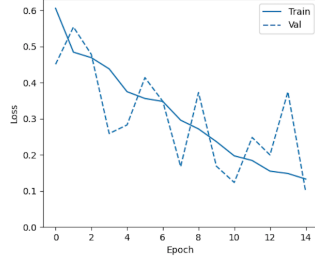


Figure 6: Loss

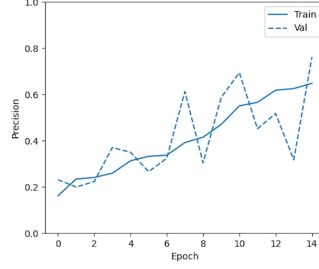


Figure 7: Precision

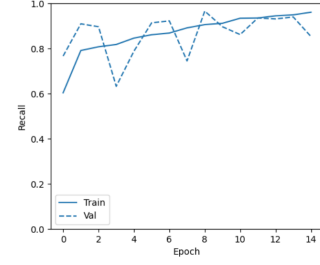


Figure 8: Recall

The confusion matrix and the plots for loss, precision and recall in figures 6-9 show the results from training and running the model on 256 batch size, with 15 epochs. This configuration was chosen for discussion and presentation, as it provided consistent satisfactory results. From the visualization of these results, it becomes evident that the model is training and learning properly, as the metrics behave somewhat as expected for a functioning model.

5.2 Discussion

As previously mentioned, there are numerous evaluation metrics to be considered when evaluating the performance of a model performing binary classification on an imbalanced data-set. By altering the weights for the two classes based on the skewed distribution results in an overall accuracy which gives a better indication of the performance. However, it is still necessary to analyze the other evaluation metrics, namely recall and precision.

From equation (5) it is clear that precision is more focused on toward the positive class than the negative class. Precision provides a measurement on the probability of detection of the positive class. Recall, commonly known as the true positive rate, on the other hand, in some way measures the ability to distinguish between the classes. Therefore, with a data-set consisting of mostly negative samples, precision is a more relevant metric when evaluating the model performance.

In the early stages of the development of the model, the default batch size was set to 32. However, when working with an imbalanced data-set such as this, it is important to ensure that each batch has a good chance of containing sufficient amounts of positive samples. After some experimentation and testing, a batch size of 256 was chosen, as this was found to yield promising results. This conclusion was deemed valid after analyzing the results in tables 1 through 6. Three different variations of number of epochs were tested, namely 10, 15 and 20. As one could clearly observe from table 2,4 and 6 in sub-section 5.1, the training loss decreases when jumping from 10 to 15 epochs, and increases when further increasing the number of epochs from 15 to 20. Considerable improvement could also be found in the precision value in the case of 15 epochs, along with a slight improvement in ROC-AUC and overall accuracy.

With regards to the model architecture there are several changes that could be beneficial in getting better results. Adding padding to the convolutional layers would mean that border pixels would be captured by the filters. Without prior knowledge of the typical input image structure it is hard to conclude whether or not this would have a considerable impact on the model's performance to capture important features. The kernel size in especially the first convolutional layer could also, even though it is usually set to 3x3, have been increases to e.g. 5x5. It is hard to tell how big impact these alterations could have on the results, but it is worth mentioning.

Additionally, the number of convolutional layers in the model could have been increased, even though more than two layers are typically only required with highly complex tasks, or excessively large amounts of training data. Finally, regarding the most important hyperparameter, different variations of learning rates should most definitely have been experimented on. We decided to use the default value of $lr=0.001$ in our optimizer. This represents a good starting point, but further tuning could potentially have yielded better results.

6 Further Work and Improvements

Several improvements, considered time-consuming and complex, could potentially be implemented in order to further the functionality and improve the effectiveness of our solution.

As briefly mentioned previously, another crucial component of a real-life viable solution would be the functionality of locating the classified brick kilns automatically. This is something that has previously been done, by teams such as those mentioned in section 2. To develop this solution, however, was considered out of scope of our intentions, due to limitations in workload. This is definitely worth mentioning as a possible improvement to our solution in the future, should we continue to develop it.

Another improvement to be made would be to perform further, and more comprehensive, parameter studies for the model. Fine-tuning the configuration of the model was not the focal point of our attention, but rather to implement a solution deemed satisfactory, before incrementally improving on the solution. Therefore, for further development, the configuration and parameters of the model could be subject to studies, in order to develop a better performing solution.

If the model was to be further developed and improved, the results would be highly likely to benefit from the entire data-set being utilized, as compared to the 40% sub-set being used for the current model. As mentioned, due to limitations in computational power and memory, a sub-set of 40% of the provided data-set was used, in order to meet these limitations. If the model was to become more complex, increasing the number of samples for the network to train on would likely be a benefactor in improving the overall performance of the model.

In addition to this, introducing image augmentation to the data-set would yield the biggest immediate improvement to the results. In augmenting the images, the network is exposed to more volatile and challenging data in the training, making for a more robust network, with better classification capabilities. The images themselves can be augmented, in order to produce semi-new, and various samples for the network to train on, and also, most importantly, more positive samples could be introduced to the training-set, in order for the model to be able to train on more positive classifications. This would be the first improvement to implement in future iterations.

More complex models could likely be more specifically tailored to the exact needs and challenges of this task, but due to limitations in time and work, as well our own lack of experience and knowledge regarding such implementations, a decision was made to aim our attention at developing a more simplistic, yet feasible, model.

7 Conclusion

The results obtained from the developed model were deemed satisfactory in regards to the estimated and expected performance towards classifying the data, and support the notion that machine learning methods are a useful tool for applications within satellite footage classification. The developed model, although simplistic, was able to correctly identify roughly 84% of the pictures containing a brick kiln. If the solution was to be further developed and improved, it stands to reason to believe that the results would improve further, thus emphasizing its utility.

Comparing the results directly and specifically against former implementations and solutions to this task is somewhat futile, as the scopes and ambitions of the task varies from research to research, but we feel satisfied in having reached, what we determine to be, promising and useful results. Our ambitions were to explore the task by developing a functioning neural network, and due to our limited former experience within this field, we aimed at using this task mainly as a learning experience as to what is required for developing such solutions.

Due to the aforementioned reason, the theoretical foundations of this report are probably, to the experienced reader, slightly shallow and possibly vague. Our main focus from the beginning was directed toward the process as a whole, and our goal was to provide a working solution, rather than being fixated on specific details, which rather serve as a foundation for further improvements. We enjoyed the project thoroughly, and found it highly interesting, useful, and challenging.

Bibliography

- Brick Kiln Image* (n.d.). <https://www.science.org/content/article/researchers-spy-signs-slavery-space>. Accessed: 2021-18-11.
- Christopher Yeh, et al. (2021). *SustainBench: Benchmarks for Monitoring the Sustainable Development Goals with Machine Learning*. <https://openreview.net/forum?id=5HR3vCylqD>. Accessed: 2021-14-11.
- Data-set* (n.d.). https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_S2_SR#bands. Accessed: 2021-11-11.
- Jihyeon Lee, et al. (2021). *Scalable deep learning to identify brick kilns and aid regulatory capacity*. <https://www.pnas.org/content/pnas/118/17/e2018863118.full.pdf>. Accessed: 2021-13-11.
- Learning NN model* (n.d.). https://www.tensorflow.org/tutorials/images/transfer_learning. Accessed: 2021-16-11.
- Metrics for imbalanced data-set* (n.d.). <https://towardsdatascience.com/what-metrics-should-we-use-on-imbalanced-data-set-precision-recall-roc-e2e79252aeba>. Accessed: 2021-14-11.
- SustainBench GitHub* (n.d.). https://sustainlab-group.github.io/sustainbench/docs/datasets/sdg13/brick_kiln.html. Accessed: 2021-11-11.
- UN (n.d.). *United Nations Sustainable Development Goals*. URL: <https://sdgs.un.org/goals>. (accessed: 11.11.2021).
- Usman Nazir, et al. (2021). *Kiln-Net: A Gated Neural Network for Detection of Brick Kilns in South Asia*. <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9115879&fbclid=IwAR1L9xRTJtpjmHBPmyVTYVQBrDxOBjtXyuoVBI5yjMKnX4ux7vm3i7r18eQ>. Accessed: 2021-14-11.