1 Introduction

For this project, we will be diving into two main related areas within machine learning, one being a problem on regression and the other on image classification and segmentation. On the first part we are going to solve problems of outliers and for model performance comparison, while on the second part, the problems are related with the imbalance of data and choice of architecture of the models to best fit the recognition and localisation of Mars' craters in satellite images.

2 Methodology

2.1 Multiple linear regression with outliers

For this problem we want to verify a linear relationship between independent variables $\{x_i \mid i \in \{1, 2, ..., 5\}\}$, corresponding to daily averages of air temperature, water temperature, wind speed, wind direction and illumination respectively, and the dependent variable y, corresponding to the concentration of toxic algae in water samples. In order to do this we want to obtain a linear model of the form

$$y = \beta_0 + \sum_{i=1}^{5} \beta_i x_i. \tag{0}$$

To estimate the parameters $\{\beta_i \mid i \in \{0, 1, ..., 5\}\}$ we perform linear regression with regularization techniques, more specifically we consider ridge regression and lasso regression. Before performing the regression we split our data set randomly into a training and a validation set 80/20.

To choose the regularization parameters λ_{ridge} and λ_{lasso} resulting in the lowest sum of squared errors (SSE) we consider a wide range of candidates and use leave-one-out cross validation, since our data set is rather small. When the best regularization parameter is found for both ridge and lasso regression we choose the one of the two models that resulted in the lowest SSE.

However, the dataset is affected by instrument noise and human error. Since human errors result in larger errors and affect the dependent variables in approximately 25% of the samples, we remove approximately the 25% most outlying samples before performing ridge and lasso regression on the rest of the data set. To do this we fit a linear model to the full dataset using least squares regression, use this model to predict the dependent variables, and then inspect and remove the samples with the largest squared error using histograms. After this we can perform ridge and lasso regression on the training set.

2.2 The ARX model

For our next problem we are given an input sequence $u(k)$ and an output sequence $y(k)$, where k is the discrete time variable. We have training samples corresponding to 2040 consecutive time stamps. Our goal is to predict an output from the input and previous outputs and inputs. The output at time k can be described by

$$y(k) + a_1 y(k-1) + ... + a_n y(k-n) = b_0 u(k-d) + ... + b_m u(k-d-m) + e(k), \tag{1}$$

where $k \geq \max(n, d+m)$, and $n, m, d \in \mathbb{N}_{\geq 0}$ are model order parameters and $e(k)$ the noise sequence. To consider this as a linear regression problem we consider the matrix form of the model equation, which is

$$Y = X\theta, \tag{2}$$

where $Y = [y(p), ..., y(N-1)]^T$, $X = [\phi(p), ..., \phi(N-1)]^T$, $p = \max(n, d+m)$, N is the number of samples in the data set, $\phi(p) = [y(p-1), ..., y(p-n), u(p-d), ..., u(p-d-m)]^T$, and $\theta = ([-a_1, ..., -a_n, b_0, ..., b_m] + e(k))^T$. By assumption $n, m, d < 10$.

Now, to train the linear model we try all combinations of integers $0 \geq n, m, d < 10$, except for combinations where $n = m = 0$, since in this case no data will be used to build the regressor matrix. While iterating over all of these combinations we do the following for each iteration:

- Construct the regressor matrix X.
- Adjust the output vector Y.
- Split the training set $\{X, Y\}$ into a new training set and a validation set using the first 75% of the samples for the new training set and the last 25% for the validation set.
- Fit three models using ridge, lasso and least squares regression and compute the SSE. The regularization parameters are chosen using leave-one-out cross validation.
- Keep track of and update the lowest SSE throughout all iterations over model order parameters, and the corresponding model order parameters and regression type.

We then choose the regression type and model order parameters that resulted in the lowest SSE, train a similar model on the original training set, and use this model to make predictions for the test set. The test set include samples corresponding to the 510 time stamps following the last time stamp in the training set. However, since we don't have the outputs in the test set, the model is excited iteratively so the predictions for previous outputs are used to predict the following.

2.3 Image classification

For this problem we want to analyse whether satellite images of Mars are showing craters or not. We want to train a binary classification model that classifies an image into class 0 if it doesn't show a crater and class 1 if it does. Our training data set consists of 2783 images of size 48 by 48, but is imbalanced and contains significantly more images with craters than without craters. In order to handle this imbalance such that the model doesn't get biased towards the majority class we experiment with data augmentation. New data in the minority class is augmented through various transformations and then added to the training set with the corresponding label. Based on the nature of the images these techniques include randomly flipping the image horizontally, vertically, rotating the image $n \cdot 90$ degrees, $n \in \mathbb{N}$, or copying the image.

In all experiments all data is scaled from $[0, 255]$ to $[0, 1]$ in order to increase numerical stability. After the data augmentation the training set is randomly split 80/20 into a new training and validation set.

We build two different types of models, a Multi Layer Perceptron (MLP) and a Convolutional Neural Network (CNN). For both model types different architectures are considered, and while training the models the F1-scores are computed. Early stopping with restoring of the best weights and automatic reduction of the learning rate is used.

Since we have access to an additional unlabelled data we tried a semi-unsupervised learning approach. More specifically we trained models using the labelled data as described above, chose the best model, predicted labels for the unlabelled data using this model, and then added the samples from the unlabelled data and the predicted labels to the training data in the cases model's prediction exceeded 95% confidence. This larger training data was then used to train the initial model again.

2.4 Image segmentation

For our last problem we are again considering images of Mars either showing craters or not, but now we want to segment the images into two classes: class 1 if a pixel is a crater and class 0 if a pixel isn't a crater, but background. We are given two training data sets, one with 48 by 48 images like in the previous classification problem, and one with 7 by 7 images corresponding to the neighbourhoods of each pixel in each image in the other data set. The 7 by 7 images are labelled with an integer 0 or 1 corresponding to the class of the pixel the neighbourhood represents, and each of the 48 by 48 images is labelled with a mask corresponding to the true segmentation of the image.

We are considering two different models, a logistic regression model and a U-Net model[1], which is a specific kind of Fully Convolutional Network (FCN). The logistic regression model is trained on the data set with the neighbourhood images, assuming that pixel values in a neighbourhood area is normally distributed, and the U-Net model is trained on the data set with 48 by 48 images. Different hyper parameters for both model types are considered and discussed in section 3.

Before training the model all images in both training data sets are scaled from $[0, 255]$ to $[0, 1]$ to increase the numerical stability. Then both data sets are randomly split 80/20 into a new training set and a validation set.

In order to handle the imbalance in the data sets we use two different approaches. In the dataset with the smaller images we use data augmentation on the new training set, but not on the validation set. This includes randomly flipping an image horizontally, vertically, rotating the image $n \cdot 90$ degrees, $n \in \mathbb{N}$, or copying the image. In the data set with the lager

[1]Ronneberger, Olaf et al.: "U-Net: Convolutional Networks for Biomedical Image Segmentation", link: https://arxiv.org/pdf/1505.04597.

images we don't use data augmentation, but define the loss function in the U-Net model as the weighted binary cross entropy instead of just the binary cross entropy. The class weights are computed from the distribution of classes in the data set. In addition the this early stopping and automatic reduction of the learning rate is used in the U-Net model.

In the end the different model types are compared focusing on the balanced accuracy in the validation sets. The results can be seen in section 3.

3 Experiments and results

3.1 Multiple linear regression with outliers

3.1.1 Outlier removal

Since this problem involves a dataset with more than two dimensions, it would be hard to visualize the hole dataset without any form of dimensionality reduction. And so, to better analyse the presence of outliers, we proceeded apply a Least-Squares method to then display in histograms and line chart it's sum of squared errors.

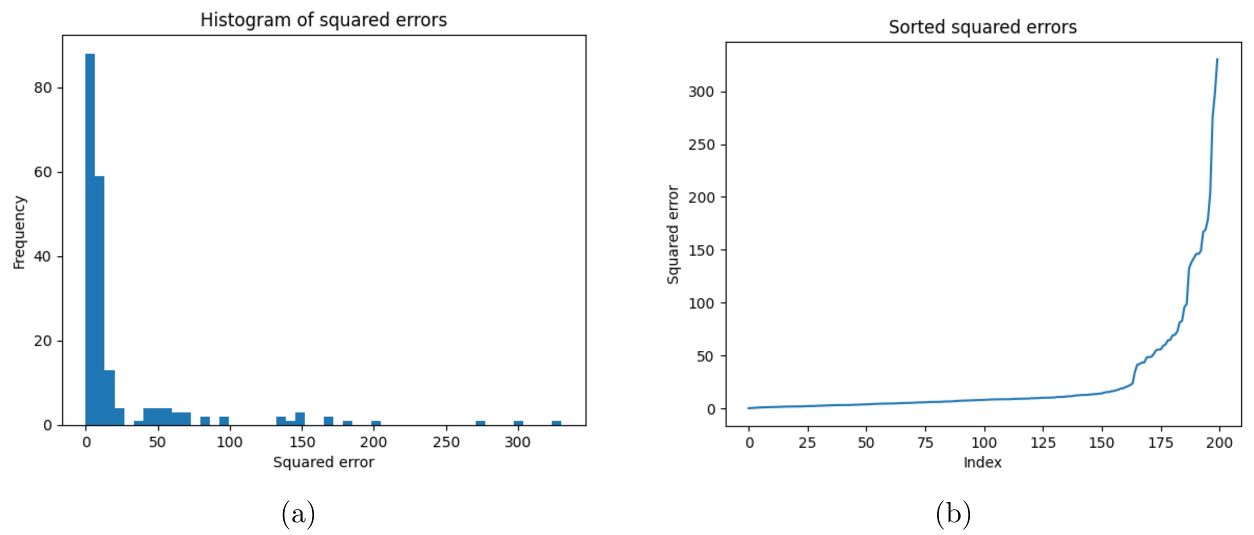


Figure 1: *In (a) we see the squared error (SE) histogram on the original data, and in (b) we see a line chart on the data sorted by SE.*

Due to the non Normal-like distribution of the results in 1a and with a clear jump in squared error from many close values around 20 to some quite sparse values from 50 and onwards on 1b. The next approach would be to set an acceptable threshold for the value of the square error of 25, meaning that all points with a squared error higher than 25 would be considered as an outlier.

After analysing the result of this approach, due to the disparity in ratio between the vast majority of the data and some last points, further thresholds of 5 and then 0.2 were decided, leading to the one last visible representation of the actual data without outliers in figure 2.

3.1.2 Model performance and comparison

In order to compare the performance of the different approaches, we decided to apply a data split where 20% of the data, after removing the human error outliers, would serve as a validation set, to compare the performance of each regressor. It is important to notice that both models applied their cross-validation algorithm and were validated on the same record distribution.

The performance from both models and their best parameters were:

- $\alpha_{\text{ridge}} = 660.362$
  with a validation SSE of 0.00744,
- $\alpha_{\text{lasso}} = 10^{-5}$
  with a validation SSE of 0.00759.

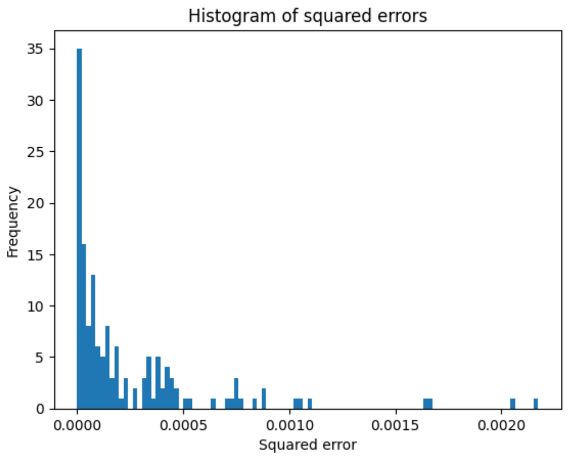


Figure 2: *Final histogram of the SE after removing the outliers.*

Analysing the results, both models were quite accurate with very close and low SSE values. The final SSE value for the Ridge regressor was 0.03059 on the independent test set, which is higher than the train and evaluation sets, but still a quite low error value.

3.2 The ARX model

To do a better critic analysis of the resulting parameters, we need to first understand what each one represents. By inspection we can conclude that the n and m can represent a window size of the past data, this is the amount of previous information that the classifier gets in order to do the current prediction. Finally, the d parameter can be seen as the delay in the input array, in theory we would think that the best value for it would be 0, but since the input values are quite erratic, changing from 5 to -5 with no values in between, we will see that adding a delay to the prediction information end up with better results.

With that said, it is natural to think that a bigger window of information would lead to better predictions, which in fact happened. In practice it was visible that for each iteration with a higher n, the best model would preform better than the previous best with lower values of m and proceeded to be overruled by the higher values of m, i.e. it was visible that higher values of n and m were minimizing the error rate of the best model. Relatively to the d parameter, the models were quite fast to stagnate on a best value, which approached 6.

The final model, least squares regressor with parameters $n = 9$, $m = 9$ and $d = 6$ predicted the outputs for the independent test set with an SSE of 6.732, which is a rather good result, when comparing to the SSE value of the predicted outputs over the validation set for the model, which was 8.002.

Now we want to check the stability of our solution. If we consider equation 0 we see that the characteristic equation can be derived from the autoregressive part of the model as

$$1 + a_1 z^{-1} + a_2 z^{-2} + ... + a_n z^{-n} = 0 \Leftrightarrow z^n + a_1 z^{n-1} + a_2 z^{n-2} + ... + a_n = 0, \qquad (3)$$

where $z \in \mathbb{C}$. We can determine the stability of our solution by extracting the coefficients $a_1, ..., a_n$ and solve equation 3. If for all roots z_r we have $|z_r| < 1$, then our solution is stable. This is fortunately the case for the least squares regressor with $n = m = 9$ and $d = 6$.

3.3 Image classification

3.3.1 Model architecture

Among the model architectures of the MLP that were tried the one that resulted in the best performance was:

1. Input layer,
2. dense layer with 128 neurons and ReLU activation function,
3. 30% dropout,
4. dense layer with 64 neurons and ReLU activation function,
5. 30% dropout,
6. dense layer with 32 neurons and ReLU activation function,
7. output layer with Sigmoid activation function.

Among the model architectures of the CNN that were tried the one that resulted in the best performance was:

1. Input layer,
2. convolutional layer with 32 kernels of size 3×3 and ReLU activation function,
3. max pooling layer with kernel of size 2×2,
4. 50% dropout,
5. convolutional layer with 64 kernels of size 3×3 and ReLU activation function,
6. max pooling layer with kernel of size 2×2,
7. 50% dropout,
8. flattening of the image,
9. dense layer with 64 neurons and ReLU activation function,
10. 30% dropout,
11. output layer with Sigmoid activation function.

3.3.2 Model performance

The results of the experiments can be seen in table 1. We see that generally the MLP performs worse than the CNN with F1 scores of 0.7663 and 0.6918 against 0.8905 and 0.9123, without and with data augmentation, respectively. Additionally, it can be seen that the MLP performs better without data augmentation, where the opposite is seen with the CNN. However, the difference is very small for the CNN.

Model	Best F1-score
MLP, no aug.	0.7663
CNN, no aug.	0.8905
MLP, with aug.	0.6918
CNN, with aug.	0.9123
CNN, with aug. and unlab. data	0.8954

Table 1: *Performance of MLP and CNN models with the optimal model architecture on the validation set. The models has been validated with and without data augmentation and semi-unsupervised learning using unlabelled data.*

Since the CNN model with data augmentation performed the best with an F1 score of 0.9123 we chose to investigate this model futher by utilizing the unlabelled data set as described in section 2.3. This semi-unsupervised learning approach combined with the CNN model with data augmentation resulted in an F1-score on the validation set of 0.8954. So, the model performance was reduced.

In conclusion the model that performed best on the validation set was the CNN with data augmentation. However, when tested on an independent test set this model got an F1-score of only 0.8282.

3.4 Image Segmentation

3.4.1 Logistic regression model

Deciding the hyper parameter C, the inverse of regularization strength, for the logistic regression model we used the function `RandomizedSearchCV` from `scikit-learn`. Given a list of candidates for C the function randomly tries a sample of the candidates and uses 5 fold cross validation on the training set to validate models using the parameters. L2 regularization is used. After finding the best model we compute the balanced accuracy on the validation set. Now, the list of candidates for C is modified and centered around the current best value of C, and the we repeat. This process is repeated until the balanced accuracy score stagnates. Note that we chose a sample of the same size as the number of candidates for C in the list, so essentially we did an exhaustive search and could have used the funcition `GridSearchCV`, also from `scikit-learn`. We started out with the candidates $0.001, 0.01, 0.1, 1, 10, 100$ and 1000 for C, and in the end we concluded $C = 0.004012$ was the best choice, with the best balanced accuracy of 0.7268 in the cross validation. However, on the validation set the balanced accuracy was only 0.5599.

3.4.2 The U-Net model

Setting up the U-Net model we also tried different hyper parameters. The overall model architecture that we are considering is the following:

1. Input layer,

2. n downsampling blocks each containing convolutional layer(s) with ReLU activation function followed by max pooling layer with stride $(2, 2)$,
3. one bottleneck block: Convolutional layer(s) with ReLU activation function,
4. n upsampling blocks where we first have a upsampling layer with stride $(2, 2)$, then concatenate the resulting image and the image returned by the corresponding convolutional layer in the downsampling block, and then new convolutional layer(s) with ReLU activation function,
5. output layer.

We experimented with different numbers of downsampling and upsampling blocks, corresponding to $n = 1, 2, 3, 4$, and whether we should use one or two convolutional layers within each of the downsampling and upsampling blocks. When we used only one of each block, we had 32 kernels of size $(3, 3)$ in the convolutional layer(s), and when we used more blocks the convolutional layer(s) in the additional blocks had 64 kernels of size $(3, 3)$. In the bottleneck block the convolutional layer(s) had 128 kernels of size $(3, 3)$. Results can be seen in table 2.

Model index	Layers	Best balanced accuracy
1	1×1 DS, 1×1 BN, 1×1 US	0.7187
2	2×1 DS, 1×1 BN, 2×1 US	0.7852
3	3×1 DS, 1×1 BN, 3×1 US	0.7974
4	4×1 DS, 1×1 BN, 4×1 US	0.7988
5	1×2 DS, 1×2 BN, 1×2 US	0.7690
6	**2×2 DS, 1×2 BN, 2×2 US**	**0.8075**
7	3×2 DS, 1×2 BN, 3×2 US	0.7999
8	4×2 DS, 1×2 BN, 4×2 US	0.8009

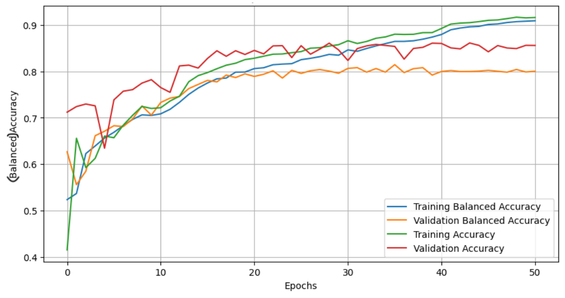
Table 2: *Best balanced accuracy on the validation set for different model architectures. DS, BN and UP stands for downsampling, bottleneck and upsamplning block(s), respectively. For instance, 3×2 DS stands for 3 downsampling blocks with 2 convolutional layers in each block.*

We can see from table 2 that model 1, 2 and 5 performed slightly worse than the others, which all have a balanced accuracy on the validation set very close to 80%. This indicates that the models with only single blocks might be too simple. However, the balanced accuracy doesn't improve with with increasing model complexity. In the contrary, one could fear that very complex models would be more likely to overfit on the training data. Since model 6 has the best balanced accuracy and also isn't to complex of a model we chose to investigate it further adding dropout layers of different ratios after all convolutional layers, except right before the output layer. The results can be seen in table 3. We see that 50% dropout might be too high of a rate, since the balanced accuracy was reduced from 0.8075 to 0.7854. The 30% dropout also reduced the balanced accuracy, but on very little. As for 40%, 20% and 10% dropout all of these rates increased the balanced accuracy, but all with under 1%.

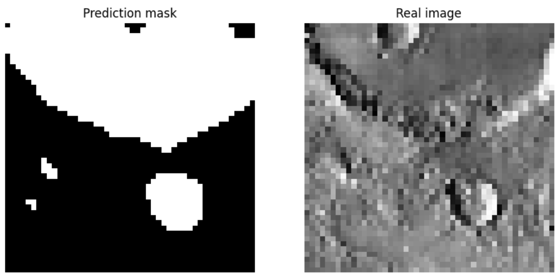
Model index	Layers	Dropout	Best balanced accuracy
9	2×2 DS, 2×2 BN, 2×2 US	0.5	0.7854
10	**2×2 DS, 1×2 BN, 2×2 US**	**0.4**	**0.8143**
11	2×2 DS, 1×2 BN, 2×2 US	0.3	0.8055
12	2×2 DS, 1×2 BN, 2×2 US	0.2	0.8124
13	2×2 DS, 1×2 BN, 2×2 US	0.1	0.8122

Table 3: *Balanced accuracy on the validation set when certain rates of dropout are added as described on page* 6 *to the model with the overall architecture* 2×2 *DS,* 1×2 *BN,* 2×2 *US.*

In conclusion it must be considered uncertain whether and to which degree dropout improves the model, however model 10 with 40% dropout was overall the best model of all with a balanced accuracy of 0.8143. The evolution of both accuracy and balanced accuracy on the training and validation set for model 10 can be seen in figure 3a. It is clear that accuracy is in general a bit higher than the balanced accuracy indicating that the imbalance in the data set might still be present although class weights were used. An example of a predicted mask can be seen in figure 3b.



(a)



(b)

Figure 3: *In (a) the performance of model 10 on training and validation set in show, and in (b) we a predicted classification mask for an image in the independent test set on the left and the actual image on the right.*

4 Discussion

For our first regression problem, see section 2.1 and 3.1.2, we concluded that the ridge regressor performed the best, with a very small margin to the lasso regressor, on the validations set. However, the SSE on the test set was significantly larger. This is might be caused by the small size of the validation set, 30 samples, compared to the larger test set of 200 samples. Although the model parameters were chosen using leave-one-out cross validation on the training data, the validation set is still so small that the higher performance might be a coincidence. Our ARX model with parameters $n = m = 9$ and $d = 6$ performed quite well on the test set, even better than on the validation set. Something to consider could be the fact that when splitting the training set into a new training and a validation set we didn't skip any timestamps in between. This means some of the data in the training set will appear in the validation set, just in different positions in the matrix X in equation 2, since we use previous outputs when we predict a new one. Similarly, is also the case when the functions `RidgeCV` and `LassoCV` perform cross validation. this issue could pose a risk of overfitting, although it only affects a few of the samples. Furthermore, we could have considered standardizing the data in both regression problems in order to possibly increase numerical stability.

Regarding our image classification problem we experienced a significantly worse performance of the CNN on the independent test set than on the validation set, see section 1. This is probably caused by the fact that we used data augmentation on our training set before splitting it into the actual training and validation set. This means that the same image can appear in the training and validation set, either as a copy, flipped or rotated. Hence, the model will have overfitted to both the training and validation set. Therefore, the augmentation should have been made only on the new training set after the data split like in the image segmentation problem, see section 2.4. Note that although we combined a softmax activation function in the output layer with the binary cross entropy loss function in the submitted code, we changed this before writing the report and used a sigmoid activation function instead. The change didn't really make a difference, so the used library probably handled the issue. Furthermore, we could have experimented with more model architectures for both the CNN and MLP, used cross validation instead of a data split considering the relatively small data set, or initialised kernels that we thought would be more likely to detect circles. Another idea for model improvement could be augmenting new data to the training set not only for the minority class, but for both classes while still balancing them. This could create even more variety. Regarding the image segmentation problem different number of kernels and kernel sizes could be investigated for the U-Net model, and further data augmentation could be used for the logistic regression. Cross validation could be investigated further for both model types. One should note that the logistic regressor performs especially poorly compared to all other models we investigated in this report. Part of this should be explained by the fact that it doesn't utilize any spacial information, which is particularly problematic when dealing with images. Another approach could be developing a CNN to classify each of the neighbourhood images in order to classify the pixel it represents.

5 Conclusion

Overall we have obtained models that performed quite well, with balanced accuracies and F1 scores above 80% for our image analysis and low SSE values for our regression problems. A clear tendency for our image analysis is the fact that convolutional networks unsurprisingly perform the best, most likely due to the utilization of temporal information in images. However, for all of our model types there are room for improvement, investigating model architectures to vary and imbalance data sets and fine tuning hyper parameters. Overall, this project worked well in order to study many use cases on machine learning and can also represent some other of possible applications in the real world, from a sleep quality tracker as a regular regressor, the automatic system for the window blinds at home as an ARX, or even the object detection on a security camera, it could merge the it's time-series characteristics with an image classifier or locator.