F3

Faculty of Electrical Engineering
Department of Cybernetics

Bachelor's Thesis

Agricultural Crop Classification from Multi-Spectral Satellite Data

David Bradshaw

May 2023

Supervisor: Ing. Jan Čech, Ph.D.

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Bradshaw David** Personal ID number: **491971**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Agricultural Crop Classification from Multi-Spectral Satellite Data

Bachelor’s thesis title in Czech:

Klasifikace zemědělských plodin z multi-spektrálních satelitních dat

Guidelines:

Agricultural crops have a typical multi-spectral signature and can be classified from satellite data [1]. Satellite data are available for the region at regular intervals during the growing season, except when the area is covered by clouds. Design a classifier to predict the class for a time sequence of multispectral responses. Use publicly available dataset [2] - training subset to learn the classifier, testing subset to evaluate it. Carry out an experiment training on (1) the full (year-long) time sequences, and (2) on shorter sequences, i.e. in case some data are not available. Compare both models on the test set. In addition to the classification, endow the predictor with a confidence score that provides a reject option. Experiment with several options for confidence scores and evaluate them using the coverage-accuracy plot on the test set.

Bibliography / sources:

[1] G. Weikmann, C. Paris and L. Bruzzone, "TimeSen2Crop: A Million Labeled Samples Dataset of Sentinel 2 Image Time Series for Crop-Type Classification," IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 14, pp. 4699-4708, 2021, doi:10.1109/JSTARS.2021.3073965.
[2] TimeSen2Crop dataset download, https://zenodo.org/record/4715631
[3] Y. Geifman and R. El-Yaniv, “Selective classification for deep neural networks,” in Proc. NIPS, 2017.
[4] Y. Gal and Z. Ghabramani, “Dropout as a Bayesian approximation: representing model uncertainty in deep learning,” in Proc. ICML, 2016.
[5] Vaclav Voracek, Vojtech Franc, Daniel Prusa, “Optimal strategies for reject option classifiers”. Journal of Machine Learning Research, 2023, in Press.

Name and workplace of bachelor’s thesis supervisor:

Ing. Jan Čech, Ph.D. Visual Recognition Group FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **07.02.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

Ing. Jan Čech, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

. | Date of assignment receipt

Student’s signature

Acknowledgement / Declaration

I would like to sincerely thank my supervisor Ing. Jan Čech, Ph.D. for giving me an opportunity to work with him. His guidance was the reason why I managed to finish my thesis in the first place, since he was always available to me and helped me with whatever I needed at that time.

I also extend my gratitude to my friends and family for their support.

I declare that the presented work was developed independently and that I have listed all sources of the information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, May 20, 2023

.....

Abstract / Abstrakt

The thesis presents results of our convolutional neural network model for classifying multispectral satellite data. We designed and trained a binary classifier for recognizing corn (maize) from other crops. The model was trained on a dataset of multispectral responses at a given location over the duration of a year. The model was designed to accept an incomplete time series (not necessarily year-long) as an input, and provides a confidence score besides the classification output. The confidence score is used for the option of abstaining from a decision in ambiguous cases. All of our experiments are carried out on publicly available TimeSen2Crop dataset presented by Weikmann et al., 2021. The accuracy, on an independent test set, of our best model for the complete input (year-long) is 96% and stays around 95% up to an input of half a year long. The accuracy can be further improved when the classifier decides only in confident cases. For example the year-long accuracy can be increased up to 99.3% by abstaining from decision in 20% of samples.

Keywords: crop recognition, incomplete time series, confidence score

Práce prezentuje výsledky našeho modelu konvoluční neuronové sítě pro klasifikaci multispektrálních družicových dat. Navrhli a vycvičili jsme binární klasifikátor pro rozpoznání kukuřice od jiných plodin. Model byl trénován na datovém souboru multispektrálních odezev v daném místě po dobu jednoho roku. Model byl navržen tak, aby akceptoval neúplnou časovou řadu (ne nutně rok dlouhou) jako vstup a kromě výstupu klasifikace poskytuje skóre spolehlivosti. Skóre spolehlivosti se používá pro možnost zdržet se rozhodnutí v nejednoznačných případech. Všechny naše experimenty jsou prováděny na veřejně dostupném datovém souboru TimeSen2Crop prezentovaném Weikmann et al., 2021. Přesnost našeho nejlepšího modelu pro kompletní vstup (celý rok) na nezávislé testovací sadě je 96% a zůstává kolem 95% až po vstup dlouhý půl roku. Přesnost lze dále zlepšit, když klasifikátor rozhoduje pouze v jistých případech. Například roční přesnost lze zvýšit až na 99,3% tím, že se zdržíte rozhodnutí u 20% vzorků.

Klíčová slova: rozpoznávání plodin, neúplná časová řada, skóre spolehlivosti

Překlad titulu: Klasifikace zemědělských plodin z multi-spektrálních satelitních dat

/ Contents

Tables / Figures

Chapter 1
Introduction

Remote sensing [1] has been used for decades. It is used for studying the Earth's surface. The topic of my Bachelor's thesis is crop classification. An accurate classification of different types of crops was made possible in recent years due to the growth in high-resolution satellite imagery. Classifying crops from satellite imagery has many potentional applications, including yield estimation, crop stress detection and precision agriculture.

A recent work by Weikmann et al. [2] showed that multispectral satellite responses gained accurate pixel-level classification of time series. The paper presents models trained on a large annotated dataset of multispectral time series sampled using satellites in Austria. The dataset has 16 different labeled crop types for example: legumes, grassland, maize etc. The paper was comparing the classification accuracy of many different architectures.

All of these experiments were, however done on complete data (data from the whole year). This is not the most ideal option for crop recognition. A better option for crop recognition would be, if we were able to use our model at any given moment of the year. An example why this may be important would be if the crops were harvested or die during the year and so we would be unable to recognize them later on. It should be desirable for the model to recognize the crop type at any given moment of the year, since waiting for the end of year to be able to use a model is not optimal. This is one of the problems we attempt to solve in our bachelor's thesis. We trained a model, that is able to classify crops, without having data from the whole year.

The other weakness of those experiments is that they did not propose any kind of confidence score [3]. In many practical applications a confidence score is a powerful tool allowing to increase the accuracy of the model, by trading of coverage, i. e. the ratio of decided samples. We propose a couple of different types of confidence and compare them to get the one with best accuracy to coverage trade off.

A popular choice for a confidence score is the softmax response [3] of a trained network. However this approach is not possible for regression and is sometimes problematic for certain tasks, since the softmax output often overfits, therefore it does not provide a good estimate of the posterior probability.

An alternative to softmax output is the usage of test time dropout [4], where the confidence score is calculated using the (inverse) variance of the output. This kind of confidence score can be used for regression tasks. This technique is a substitute for ensemble statistics and is rather suitable for finding out-of-distribution samples than in-distribution ambiguous samples.

Franc et al. [5] had recently proposed another methodology for the rejection option. He proposes for an independent classifier to be trained for predicting errors of the original classifier. Training of this classifier is to be done on unseen validation data. During training we attempt to maximize the area under the coverage-accuracy curve to maximize the trade-off. This method is suitable for both classification and regression tasks. The downside of this method is that there is no guarantee of the confidence

predictor making reliable predictions of the original classifiers mistake. Additionally the confidence predictor may be as complex as the original predictor and may need as much data for training.

The rest of this thesis is structured as follows. In Sec. 2, we present the methods we used for our work. In this part is included, how we prepared a suitable dataset, the architecture of our network, training of our networks and calculations of our confidence score. In Sec. 3, we present our experiments and results of these experiments. In Sec. 4, we write our discussion, where we suggest alternative methods and also possible future work expanding on ours. The final Sec. 5, is a conclusion of the thesis.

1.1 Problem statement

The goal of this Bachelor's thesis is to design and train a classifier for recognizing types of crops from multispectral satellite time sequences and then propose a suitable confidence score, with the highest possible accuracy to coverage trade off.

The data is prepared from the publicly available TimeSen2Crop dataset [6]. We resample the data for a more suitable input of our network.

We then designed/adapted a network to be able to take input of the correct data size. We tested multiple training strategies, in an attempt to train a model capable of recognizing the crop type even from incomplete data (not necessarily year-long).

Lastly we measured multiple ways of calculating confidence scores and compare them to figure out, which of them has the maximal area under the coverage-accuracy curve.

Chapter 2
Method

2.1 Dataset preparation

For my work I used the TimeSen2Crop dataset [6], which is a publicly available dataset. It consists of about 1.2 million samples. The dataset was created in Austria over the span of one year , starting in September 2017 and ending in August 2018. Data from this dataset is inconsistent in it's length, and so we resampled it, since our model can only accept data with a consistent length.

The sampling was done over 15 different regions, which are seen in Fig. 2.3. Over a single year, each region was scanned periodically every 5 days, though the data is often missing, which happened when there were either clouds or snow covering the crops. Furthermore each region was scanned on different days, leading to a phase shift of up to 4 days (considering it being one directional).

One pixel has a spatial resolution ranging from 10 to 60 meters. Each pixel can be made of a different amount of scans, but the number and dates of scans are consistent throughout each region. Each of these scan contains data consisting of 9 multispectral responses and a flag regarding the condition of the pixel (0 - clear, 1 - cloud, 2 - shadow, 3 - snow), together with a timestamp of the date of acquisiton. Each pixel is anotated with one of 16 different labels. We wanted to design a simple binary classifier only designed for recognizing maize (corn), so for our labels we only used: 1 - maize, 0 - all the other labels.

We resampled the data into a matrix of fixed size 10×73, where 10 is the number of data in a single scan and 73 is for a whole year with a step of 5 days ($365/5 = 73$), i. e. we created our data by 'squeezing' them into a shorter length. We used the nearest neighbor interpolation, where before interpolating I filled in the missing data with zeros. The nearest neighbor interpolation creates an error by shifting data, which could create a drop in the accuracy of a model. Since for us the step of scanning is only five days, by using the nearest neighbor interpolation we have a maximal shift/error of 2 days, (seen in Fig. 2.1), which we can consider a small enough error, so that it does not have a big impact during training and classification, considering that during the growth a two day window will not have a big impact on the crops, when we take data from a whole year. This way we got a dataset, that is consistent in the size of its data, which is suitable for a convolutional neural network. This was also suitable for the work we were doing, moreover we could easily simulate missing/not yet existent data by setting corresponding columns to zeros (visualized in Fig. 2.2).

A visualization of our resampling method is seen in Fig. 2.1. It is a simplified diagram, since it only shows a small subset of the whole resampling. Even though it shows just a subset of the whole resampling, it shows actual data of some of the regions i. e. it shows a part of our real resampling. The representation is not done on the whole year, but just on a subset of 15 consecutive days or half a month. The first row shows the dates of each separate scan. The next three rows show distinctive data on which we proceed to show the actual resampling. The blue columns are the ones from which we extract

data into our resampled dataset. The '|' symbol shows when the samples are captured and each single symbol stands for 9 multispectral responses and a flag channel. The red arrows show, where is the data moved to after using the nearest neighbor interpolation. The bottom half shows how a final result of our resampling of the 15 consecutive days would look like and is prolonged to show how it fits into our dataset.

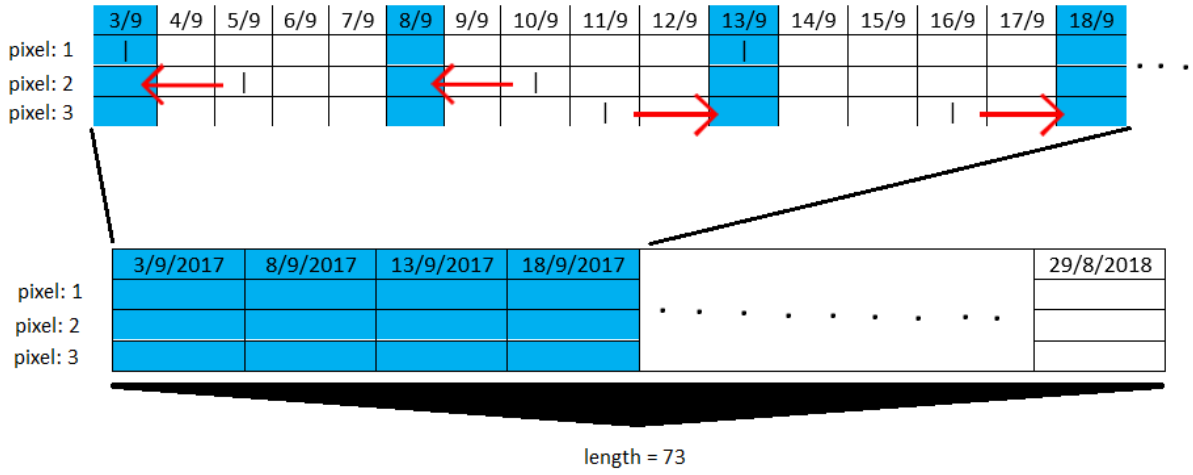

Figure 2.1. Data preprocessing of our resampling method. The representation is only done on a shorter series, we only show resampling of 15 consecutive days, our resampling was done on a whole year i. e. 365 days. The upper half is a representation of the data as was provided. The first row only shows dates of samples. The columns highlighted in blue are the dates in which we do our resampling. The next three rows show three distinctive data time series. We portray data with the symbol '|', where each of these symbols stand for one scan. Each '|' is equal to 9 multispectral responses and a flag received from one pixel. The red arrows show the resampling when using nearest neighbor. The places that would not have any data near them will be filled with zeros. The final result of our resampling is shown in the bottom half. In this part of the figure one cell is an equivalent of one scan and it either contains data or is set to zeros if the scan was not usable. The bottom half shows a whole year i. e. the exact length of data after our resampling

For our work we used the sets from TimeSen2Crop dataset [6], where both validation and testing sets are concluded exactly of one of the 15 regions. The distribution of sets is shown in Fig. 2.3. These sets consist of approximately 800 thousands training samples, 100 thousands validation samples and 300 thousands testing samples.

Another important factor is the prior distribution of maize in these datasets, which may affect the outcome of our experiments. For example testing on a dataset with a smaller percentage of maize might have higher accuracy, than when tested on a dataset with higher amounts of maize. This event might happen simply, because the model is more likely to recognize the background class, than the class of maize, not because it recognizes better. In the originally intended splitting of datasets the training dataset has 13.28% of maize, the validation dataset has 17.19% of maize and the testing dataset has 12.82% of maize.

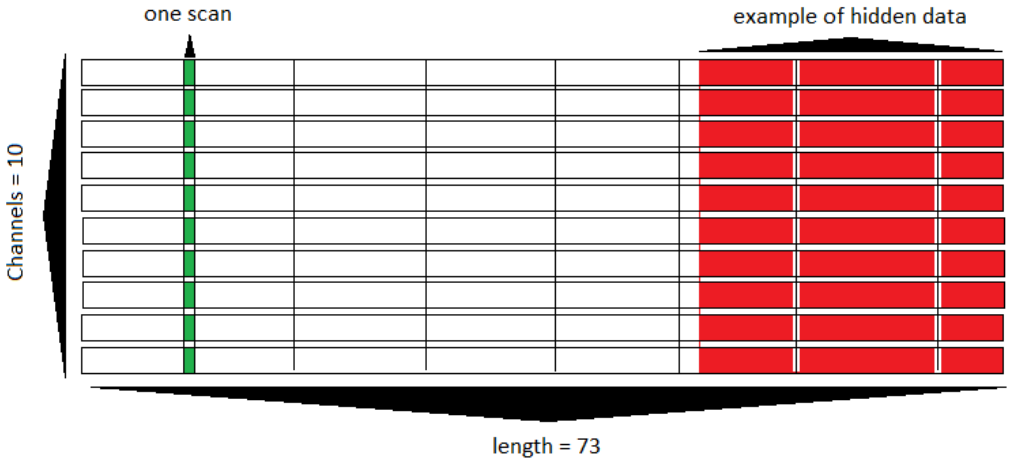

Figure 2.2. A portrayal of how one sample of a pixel may look like. We have data length equal to 73 (one scan every 5 days for a whole year), with 10 channels (9 multispectral responses of different wave lengths and the last one is a flag channel containing the condition of the pixel). An equivalent of one scan is shown in green and each scan consists of data from 10 different channels. This figure also contains a visualization of hiding data, where the end of data of the time series is set to zero. In the figure the example of hidden/zeroed out data is in red. The data is hidden from the end and is proportional to the amount we want to hide i. e. we hide all 10 channels over some part of the length.

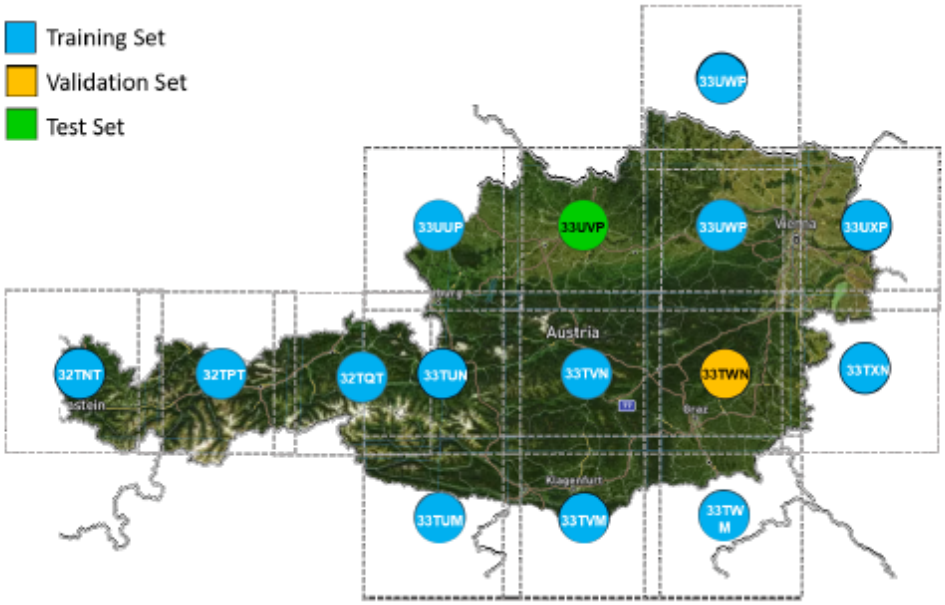

Figure 2.3. A picture taken from [2]. A map of Austria showing the distribution of the sets into training validation and test sets. In the figure we can also see how the regions are distributed throughout Austria. There are 15 different regions on the map. The regions that are highlighted in blue are regions used for the training dataset, the one highlighted in yellow is the validation dataset and the one highlighted in green is the test dataset

2.2 Network architecture

For the architecture of our network we used an adapted ResNet-50 model [7]. A visualization of our adapted network is seen in Fig. 2.4. We had to adapt the network to accept an input of 10×73, where 10 is the number of channels and 73 is the dimensions. The typical ResNet-50 uses a 2 dimensional convolution, but we had to change this to a 1 dimensional one, due to our data having only one dimension along the time series, as we set the other dimension as channels.

We replaced the first convolution layer with a convolution of kernel size 10. After which we put a max pool layer with kernel size of 3, stride of 2 and padding of 1. The middle layers of ResNet-50 were left unchanged, except for changing the dimensions to 1D. After these layers, we continue with an average pool. Next we put a linear layer. And at the complete end we put a softmax function. Each convolution is followed with a BatchNorm and ReLu for non-linearity. During training we used the binary cross-entropy loss and Adam optimizer.

2.3 Network training

In our research we trained two different networks with an exact same architecture. First we trained a baseline network trained on complete data and then a robust network trained with data augmentation explained in Sec. 2.3.2.

2.3.1 Baseline network

As previously mentioned we trained the baseline network on complete data (data from the whole year). During training of this network we went through the training set 150 times without dropout and then we went over it another 100 times with dropout engaged for better regularization. We set the dropout probability to 0.5. After these 250 epochs our training had converged. We had selected the best model based of the lowest error rate on the validation set.

2.3.2 Robust network

The difference between the baseline network and the robust one is that for the robust one we used time series data augmentation. Every batch during training we uniformly switched the data to 'unavailable', starting from a random position. This simply means that a certain amount of columns at the end was set to zero. This way we simulated an unknown future or data earlier on in the year. For a visual explanation see Fig. 2.2.

Training time for the robust network was the same as for the baseline one. We went over the training dataset for 150 epochs without dropout and then for 100 epochs with dropout turned on. The model accuracy had converged over these 250 epochs. The best model was selected based of the accuracy on the validation dataset, on which we used data augmentation in the same way as for the training dataset.

2.4 Confidence scores

In our research we have tried different types of confidence scores [3]. The first of these confidence scores is created from the softmax response. Both p and $1 - p$ are outputs of the softmax from our network. We calculate the confidence score as follows

$$c_s = 2 * \max(p, 1 - p) - 1.$$

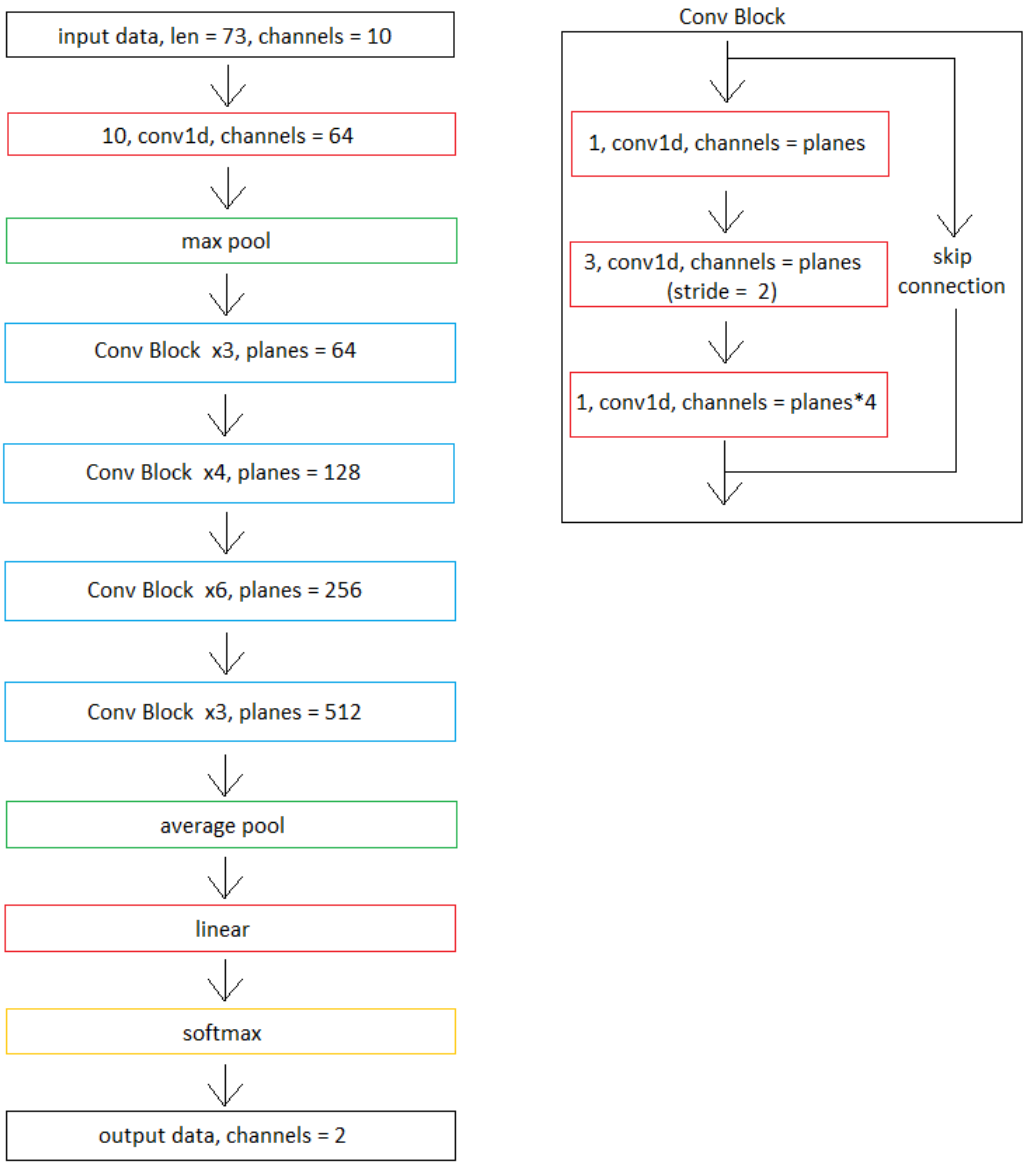

Figure 2.4. Our adapted ResNet-50 network architecture. The architecture itself is seen in the left part and the right part shows an in depth view of a Conv Block. At the beginning we have data, with length 73 and 10 channels. We proceed with a 1 dimensional convolution with kernel size 10. Then we have a max pool layer. The next 4 layers are Conv Blocks with their count (the number of times they are repeated) being [3, 4, 6, 3]. Each Conv Block consists of 3 convolutions with kernel sizes [1, 3, 1] and the middle convolution has stride = 2 for each first cycle of a count (block of the architecture on the left, which are equal to [3, 4, 6, 3]). After these blocks there is an average pool, a linear layer and a softmax function.

Since p is in a range of $p \in [0, 1]$, we will end up with a confidence also in the range of $c_s \in [0, 1]$, where 1 symbolises highest confidence and 0 symbolises lowest confidence.

Another option is calculated from using dropout during testing [4]. We will let the network run multiple times on the same input, with dropout turned on. By turning on

dropout, we introduce a certain amount of randomness into the network. The confidence score is then calculated from the variance of these different results. The equation for calculating the confidence is

$$c_d = 1 - \alpha * var\{p_1, ..., p_N\}.$$

Where p_i is one of the outputs of the softmax (for calculating variance it does not matter which one). N is the number of repeated network executions. α is a constant normalizing the confidence score, it makes the lower end of the confidence start closer to zero (Changes the confidence to be closer to the range of [0,1]), it does not change the order of the confidence score. In all our experiments we used $N = \alpha = 10$.

Last of the confidences, calculated during testing, that we used is based only on the completeness of data. It is calculated by the current length of the input. The confidence is as follows

$$c_c = L/L_{max}.$$

Where L is the length of the currently available sequence and L_{max} is the maximal length of the sequence.

2.4.1 Network for predicting confidence

The last type of confidence we worked with was a convolution neural network trained to predict confidence. For the model we used the network architecture as was described above, i. e. we used the same network architecture as for our baseline and robust model. Training of this network was done on the validation dataset. During training the network had gone over the validation dataset for 200 epochs. For validating the model during training we used a subset of the validation dataset, on which we had not performed the networks training. The best model is selected based on having the largest area under the coverage-accuracy curve over the subset of data used for validating. This model was trained without dropout ever engaged.

For training of this network we needed to use another one, that we use for obtaining labels for our confidence classifier network. We had used only our robust network, as it works better then the baseline one. During training we used the same technique of time series augmentation for simulating unknown future as for the robust network. After data augmentation we ran the input data through our robust network. We then set respective labels for our network if the robust network had estimated correctly or wrong. We used labels as: 0 - correct prediction, 1 - wrong prediction.

We then let the same exact input go through the network supposed to predict confidence. As labels to train the network we used the ones described above, that are based of the correct or wrong guess of our robust network.

A visualization of setting the labels for the training of this network is seen in Fig. 2.5.

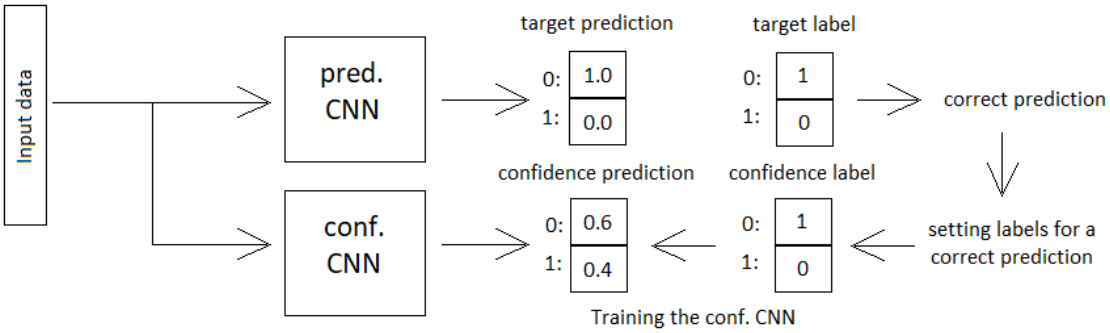

Figure 2.5. A show of training and setting labels for the convolutional neural network predicting confidence. The input data is the same for both networks. After putting the input through the robust model (pred. CNN) we get a target prediction, which we compare to the target label. In the shown case we predict 0 and the label is 0, so the robust network made a correct prediction. Since the robust network made a correct prediction we set the confidence label to 0 (label of a correct prediction), which is then used as the label for the conf. CNN. We then backpropagate, through the conf. CNN, thus training it to predict confidence of the robust network.

Chapter 3
Experiments

3.1 Training plots

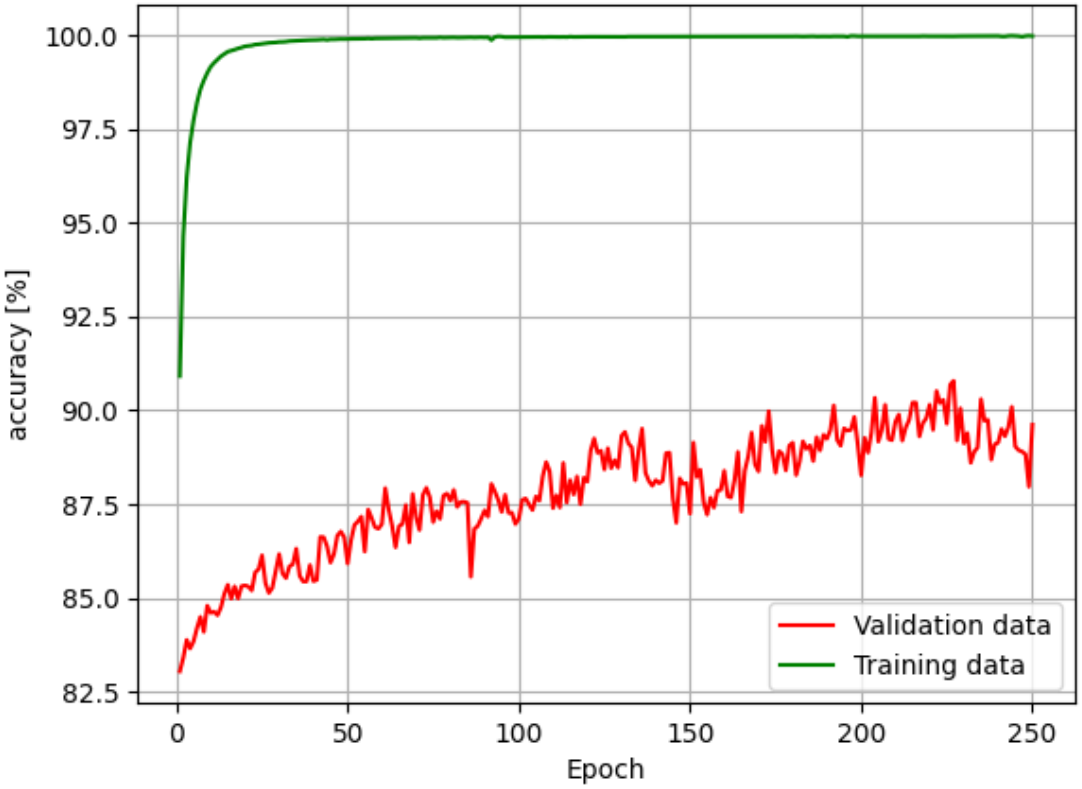

Figure 3.1. Training curves of our baseline model i. e. a model trained on complete time series. The plot shows accuracy on the training and validation dataset of a epoch during training. The horizontal axis shows current epoch during training. One epoch is for one whole rotation over the training dataset. Between epochs 0 to 150 the training is done without dropout and between epochs 150 to 250 we included dropout to our training. Accuracy on the vertical axis is the percentage of correctly classified samples.

Plots in this section show training curves of both our models, the baseline model and the robust model. Fig. 3.1 shows the training curve of our baseline model and the second Fig. 3.2 shows the training curve of our robust model.

The horizontal axis of both of these plots show the current epoch of training. The first 150 epochs are done on a network without dropout, whereas the last 100 epochs

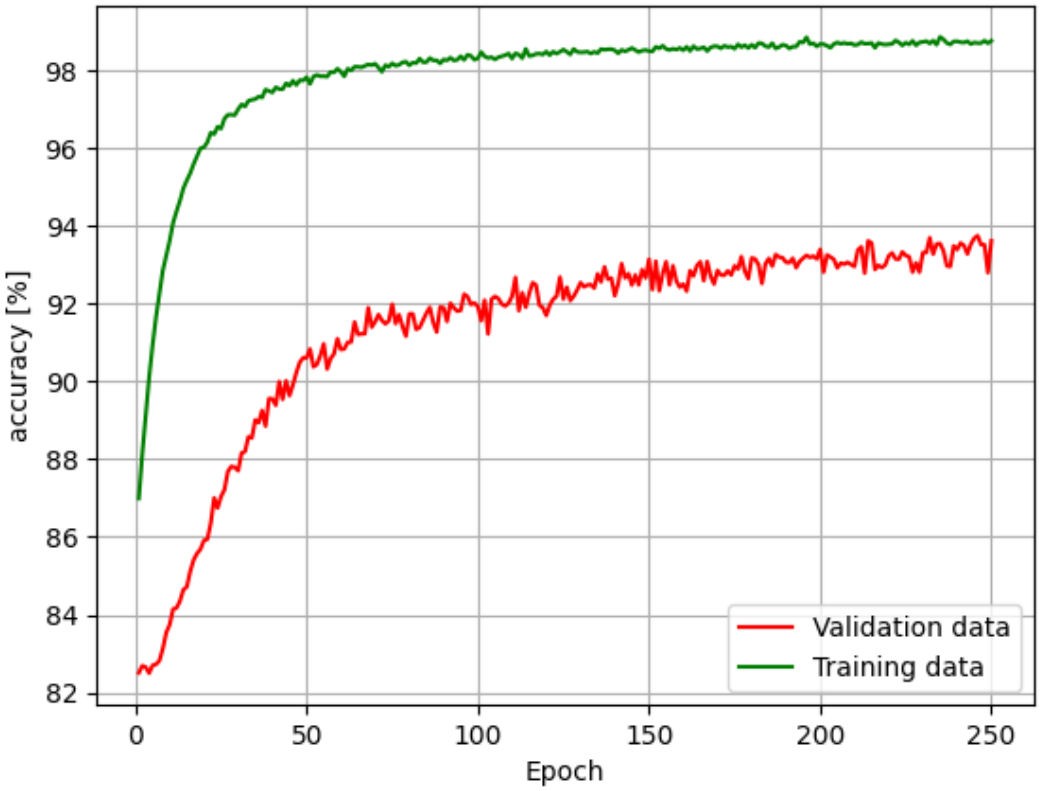

Figure 3.2. Training curves of our robust model. The robust model is trained with data augmentation i. e. uniformly shorter sequences. The plot shows accuracy on the training and validation dataset of a epoch during training. The horizontal axis shows current epoch during training. One epoch is for one whole rotation over the training dataset. Between epochs 0 to 150 the training is done without dropout and between epochs 150 to 250 we included dropout to our training. Accuracy on the vertical axis is the percentage of correctly classified samples. During training of the robust model both the training and validation dataset use data augmentation as described above.

are with dropout included in our model. The baseline model was trained on complete year long data, whereas for the robust model we used data augmentation on both the training and the validation dataset.

An important distinction of the two datasets is that the training dataset has 13.28% of maize, whereas the validation dataset has 17.19% of maize. This creates the big jump between the accuracy on these two datasets, over the whole of training and even just as the training commences.

In the training plot of our baseline model Fig. 3.1. We can see that the model does not seem to train that well. It seems that even though it trains perfectly on the training data, achieving almost 100% accuracy, it never seems to work good on the validation data. There is a visible improvement on the validation dataset, but it is visible that the model has overfitted on the training data. This observation can be seen just before the benchmarks of 150 epochs (where the training without dropout ends) and 250 epochs (end of training) the model is starting to lose accuracy on the validation data, leading me to believe it is starting to overfit.

From the training plot of our robust model Fig. 3.2, it seems that we have achieved healthy training. Both of the accuracies over the training and validation data have a good curve and there is visible correlation between the models accuracy on training and validation data. The biggest jump between them is pobably because of the difference in prior distribution of maize in the two datasets. From the training plot it seems that the model works quite good and is universal enough to work well on any data, that would be experimented upon.

When we compare the two training plots Fig. 3.1 and 3.2, we can see a big difference, between them. On the accuracy over validation data, the baseline model has a learning curve that is close to being linear and the robust model has a learning curve resembling a logarithmic one, which is an expected shape for a learning curve. Another observation that we can see is that the baseline model has bigger spikes of dropping accuracy on validation data and even at times seems to be going down constantly, where it is probably starting to overfit on the training data. In the training plot of the robust model there is visible oscillation of the accuracy over validation data, but there are no bigger spikes, where the accuracy drops. Moreover the accuracy of validation data has visible constant growth and never has a downwards tendency. All of these observations make it seem as though that the training of the robust model has gone better than of the baseline model.

3.2 Classification accuracy

We tested the classification accuracy of both of our models in two different ways. During these tests we used our testing dataset of aproximately 300, thousand samples. The testing dataset contains 12.82% of maize.

Firstly we tested both the baseline model and the robust model on complete year-long data. During this testing the data was put into the network untouched since our resampling, which is an equivalent to when we were training our baseline model, because of this, it could be expected that the baseline model might have a better performance during this test.

Secondly we tested both models with data augmentation, same as when training our robust model, i. e. we switched a random proportion of data from the end to zero. The proportion of data switched to zero was exactly the same for a whole batch. The range of possible proportion hidden is from 0% to 100%, so the network can even get an empty input (just zeros). For the purpose of making sure that the randomness factor of testing makes no difference in the results, the proportion set to zero is the same for input data of both models. Since this testing is done with data augmentation same as during training of our robust model, we could make an assumption that the robust model will classify better.

The results of these tests are shown in Table 3.1. The data presented is in percentage. In the table we can see that the robust model does significantly better achieving almost 4 percentage point higher accuracy on not only the shorter time series, but the complete time series as well, which might be against the expected result. The reason for this may be that the robust model has trained to generalize better, because by augmenting the data it has seen more variations of the same data, thus simulating a larger dataset. Even though this particular test does not signify how would the model be used in the real world, it implies that the method of training of our robust model is beneficial and has significantly improved the accuracy of our model.

	baseline model	robust model
complete time series	92.53	96.04
shorter time series	88.55	92.34

Table 3.1. Table of the test accuracy [%] of our classification models. In the columns we have our models, the first is our baseline model and the second is our robust model. In the rows we have the lengths of time series over which we got the accuracy. The complete time series row shows accuracy when using complete year-long data and the shorter time series row shows accuracy when using time series data augmentation.

3.2.1 Accuracy as a function of time series length

Another experiment we performed is measuring accuracy as a function of time series length. In the plot we can see the accuracy of different models depending on time series length.

We define the length of time series by completeness, which is plotted on the horizontal axis. In the plot 100% completeness means that the input is complete (a whole year), 50% means 6 months and 0% means an empty input. The data preparation is done in the same way as when we were training the robust network, we set a certain amount of columns at the end of data to zero. The only difference is that during training of our robust model we cut a random length for each batch, whereas for this experiment we cut a fixed amount for each point of the plot, which is tested over the whole testing dataset.

In the extreme case of completeness 0% the model decides based on the prior distribution, where during training the background class has a bigger representation, so that is what the model always predicts. A guess when the data completeness is 0% is obviously not helpful in a real world application, because even though accuracy of approximately 87% seems high in reality it does not recognize anything, but just guesses the background class for every input. This is shown for the sake of completeness.

The curves of the baseline and robust model are both tested on just one model over the whole plot, whereas the curve of the specialised models has been tested on many different models. Each of these specialised models was trained with a specific proportion of data hidden and then it was tested with that same proportion hidden. Because of the number of these models, we were only able to train them on 100 epochs over the training dataset and the training was done without ever turning dropout on. This may result in a loss of accuracy compared to the other models so we should keep that in mind. For the accuracy with completeness 100% we used our baseline model, which had more training data than the rest of our specialised models, this makes it so that the accuracy there is higher than the rest of the curve.

The results of this experiment are shown in plot Fig. 3.3. The accuracy of our baseline model does not seem to be doing very good, when given incomplete data. The accuracy drops quite quickly and under completeness 60% the recognition ability is basically zero, since the accuracy is around the prior distribution of the background class i. e. we could achieve the same accuracy by always guessing the background class.

The robust model seems to be doing very well, achieving accuracy of 95% over completeness of just around 50%, losing just 1% of accuracy while classifying from 50% of data. The accuracy after this point is dropping faster, but the model still has reasonable accuracy until 30% of completeness. In the plot we can see that the robust model is superior to the others over almost the entirety of completeness. The only place, where the other models have higher accuracy is around completeness 0%, where the

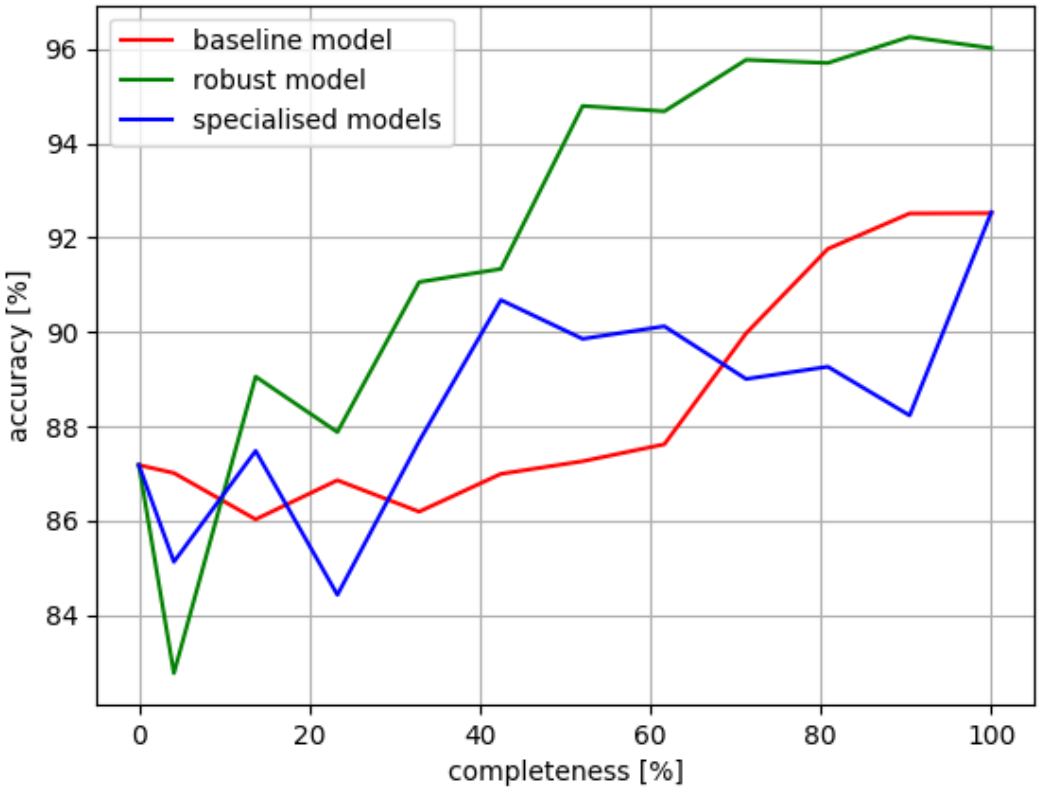

Figure 3.3. A plot of accuracy as a function of time series for our baseline model, robust model and specialised models. The specialised models are trained for an exact data length and we also test them only with that data length. Both for the baseline and the robust model this whole experiment was done using just one model, whereas for the specialised models each point in the plot is tested using a different model, trained for that specific data length. On the horizontal axis we have completeness in percentage, which stands for the length of the input. 0% completeness is for an empty input and 100% is for a whole input equal to the length of one year. Accuracy is the percentage of correctly classified samples over the whole testing dataset with the specific amount of data length.

percentage of shown data is very small. The reason for accuracy of the robust model around completeness 0% being low compared to the other networks may be, because the robust model will attempt to guess the maize class sometimes and with the lack of data often misses, whereas the other networks might just always guess the background class achieving higher accuracy. This may be substantiated by the fact that the baseline model achieves the highest accuracy, since during training it never encountered data with completeness lower than 100%.

The specialised models have a tendency of growing accuracy until 40% of completeness (except completeness 100%, where we used the baseline model with more training and dropout turned on for a part of it). The specialized models are losing to the baseline model close to completeness 100%, which is probably due to the lack of epochs during training and the fact that they were trained without dropout, since dropout helps the model with regularization. The specialised models perform better than the baseline model in the middle of the plot and they do not perform well when getting near completeness 0%. They are inferior to the robust model in most of the plot. The

reason why the specialised models do not perform well may be due to insufficient training. If the networks had the same amount of training iterations as the robust model, they would probably surpass its accuracy for some completeness, but the robust model would certainly do better for other completeness.

3.3 Confidence evaluation

We evaluate the quality of a certain confidence using the standard coverage-accuracy curve. We do this on two different datasets. First we evaluate it on complete year-long data and the second dataset we evaluate the confidence on is the dataset with shorter uniformly sampled time series. All of the tests were done using our robust model for target prediction.

The coverage-accuracy curves for all the different types of confidence on complete year-long data is shown in Fig. 3.4 and the confidence-accuracy curves for the same test is in Fig. 3.5. In these graphs C_s is the confidence curve of our softmax response, C_d is the confidence curve when using test-time dropout and C_{cn} is the response of our independent confidence classifier network.

In Fig. 3.4 we can see that both the softmax response and the test-time dropout confidence can significantly improve the resulting accuracy of our robust model by trading in coverage, i. e., abstaining from decision for some data, whereas our independently trained network for classifying confidence can improve accuracy, but it would have to trade in more coverage for a smaller boost in accuracy. Using either the softmax response or the test-time dropout confidence we can for example achieve accuracy of 99.5% for coverage 60% (abstaining from decision on 40% of data) or accuracy of 99.3% for coverage 80%. From the coverage-accuracy curve the softmax response and the test-time dropout confidence seem like they are equally good, having almost the same exact shape.

From Fig. 3.5 we can set a desirable threshold of confidence on when to decide and when to abstain from a decision. In this graph we can see that the softmax score is spread out better between the range of confidence, than the test-time dropout based confidence. This makes the softmax response confidence better suited for usage, because setting a threshold for the test-time dropout would be more difficult and the threshold would be very unstable.

The coverage-accuracy curves for all the different types of confidence on data with time series augmentation (shorter time series data) is shown in Fig. 3.6 and the confidence-accuracy curves for the same test is in Fig. 3.7. In these graphs C_s is the confidence curve of our softmax response, C_d is the confidence curve when using test-time dropout, C_c is the confidence score gotten from completeness and C_{cn} is the response of our independent confidence classifier network.

From Fig. 3.6 we can see that again the softmax response and test-time dropout are superior to both of the other confidences, since these two curves have the most area under them. Both the softmax response and test-time dropout confidence curves are very similar in their shape. Using either the softmax response or the test-time dropout confidence we can for example achieve accuracy of 99% for coverage 60% or accuracy of 97% for coverage 80%. The curve of confidence calculated from completeness has an instantaneous drop at the beginning of the graph, since the model does not have that high accuracy even when classifying from complete data and is thus not well suited for use.

From Fig. 3.7 we can set a desirable threshold of confidence on when to decide and when to abstain from a decision. In this graph we can see that the softmax score is spread out better between the range of confidence (it is more linear), than the test-time dropout based confidence. This makes the softmax response confidence better suited for usage, because setting a threshold for the test-time dropout would be more difficult and the threshold would be very unstable.

Altogether the softmax response seems like the best confidence score we tested and the test-time dropout is close second to it. One more reason why the softmax response may be better for usage is that during classification we calculate the softmax response directly from the output of our robust network, whereas to calculate the test-time dropout confidence, we need 10 outputs of a network with dropout engaged, i. e. to get the confidence score of test-time dropout we need 10× the calculation time of acquiring the softmax response.

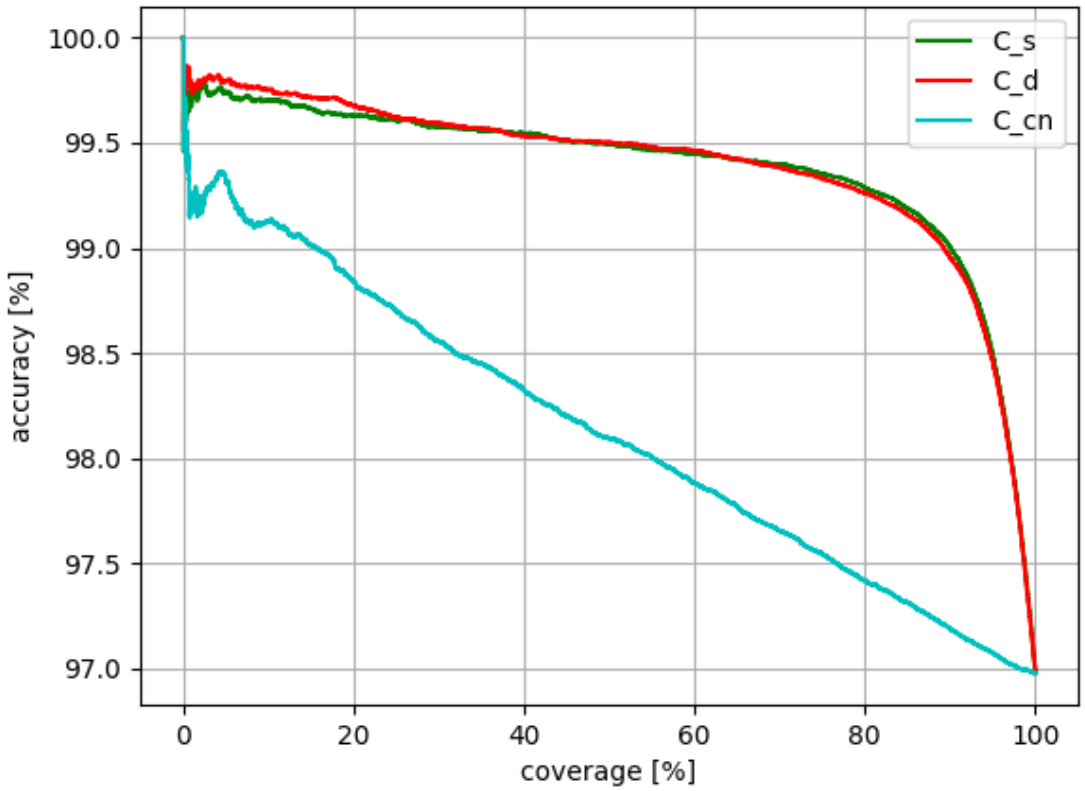

Figure 3.4. A plot showing the coverage-accuracy curve for complete year-long data using our robust model for target prediction. When preparing this plot, we order the outputs of our model based on confidence. On the horizontal axis we have coverage in percent, which represents the percentage of data shown at that point. Accuracy on the vertical axis is the percentage of correctly classified samples of data at that point in the plot. C_s is the curve for the softmax output of the model, C_d is the curve for confidence acquired from test-time dropout and C_{cn} is the curve for confidence acquired from the separately trained confidence classifier.

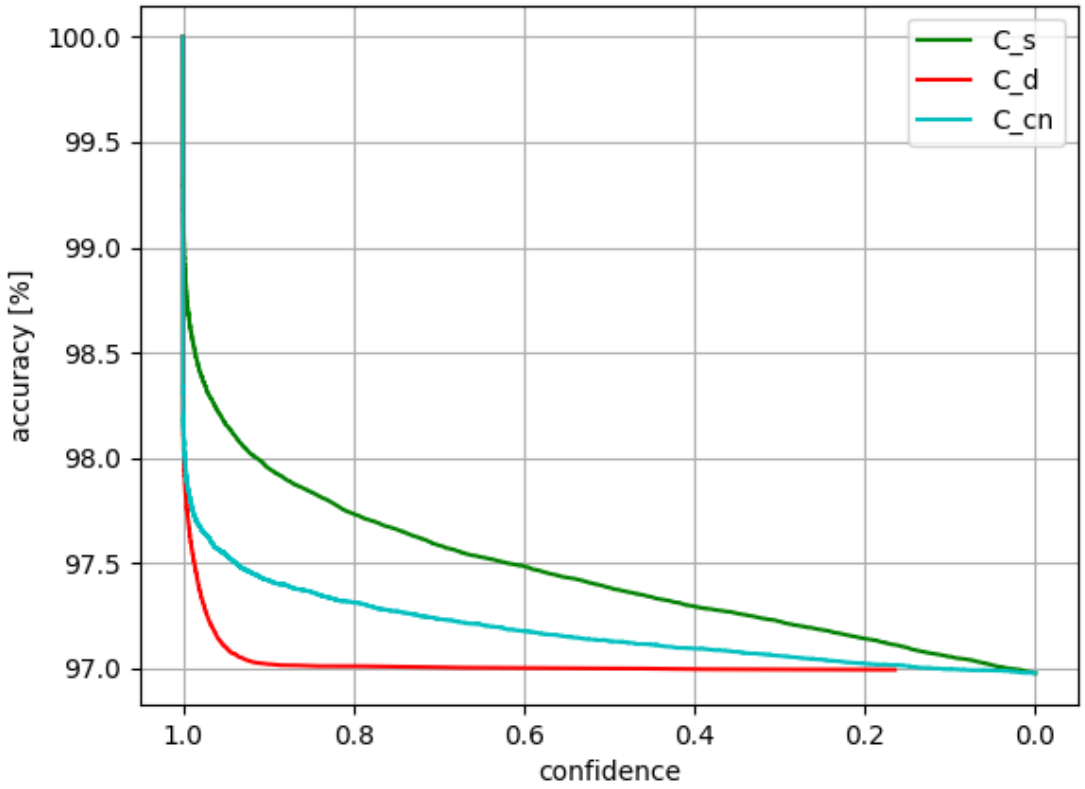

Figure 3.5. A plot showing the confidence-accuracy curve for complete year-long data using our robust model for target prediction. When preparing this plot, we order the outputs of our model based on confidence. The horizontal axis is the confidence, which was sorted in decreasing order. This plot does not have a proportional distribution of samples along the horizontal axis. The left side of the plot probably has a higher density of data. Accuracy on the vertical axis is the percentage of correctly classified samples of data, this accuracy is calculated in a cumulative way, so at each point in the plot the accuracy is calculated from current and all predeceasing data. C_s is the curve for the softmax output of the model, C_d is the curve for confidence acquired from test-time dropout and C_{cn} is the curve for confidence acquired from the separately trained confidence classifier.

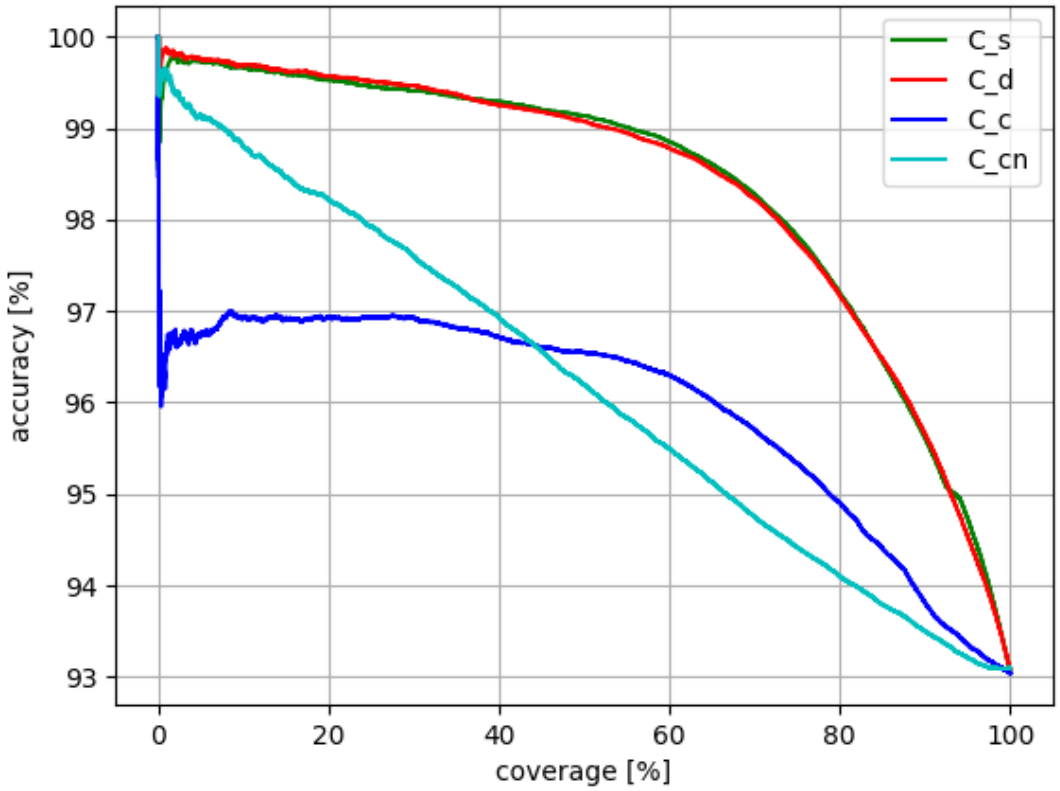

Figure 3.6. A plot showing the coverage-accuracy curve for shorter time series data using our robust model for target prediction. When preparing this plot, we order the outputs of our model based on confidence. On the horizontal axis we have coverage in percent, which represents the percentage of data shown at that point. Accuracy on the vertical axis is the percentage of correctly classified samples of data at that point in the plot. C_s is the curve for the softmax output of the model, C_d is the curve for confidence acquired from test-time dropout, C_c is the curve for confidence calculated from the completeness of data and C_{cn} is the curve for confidence acquired from the separately trained confidence classifier.

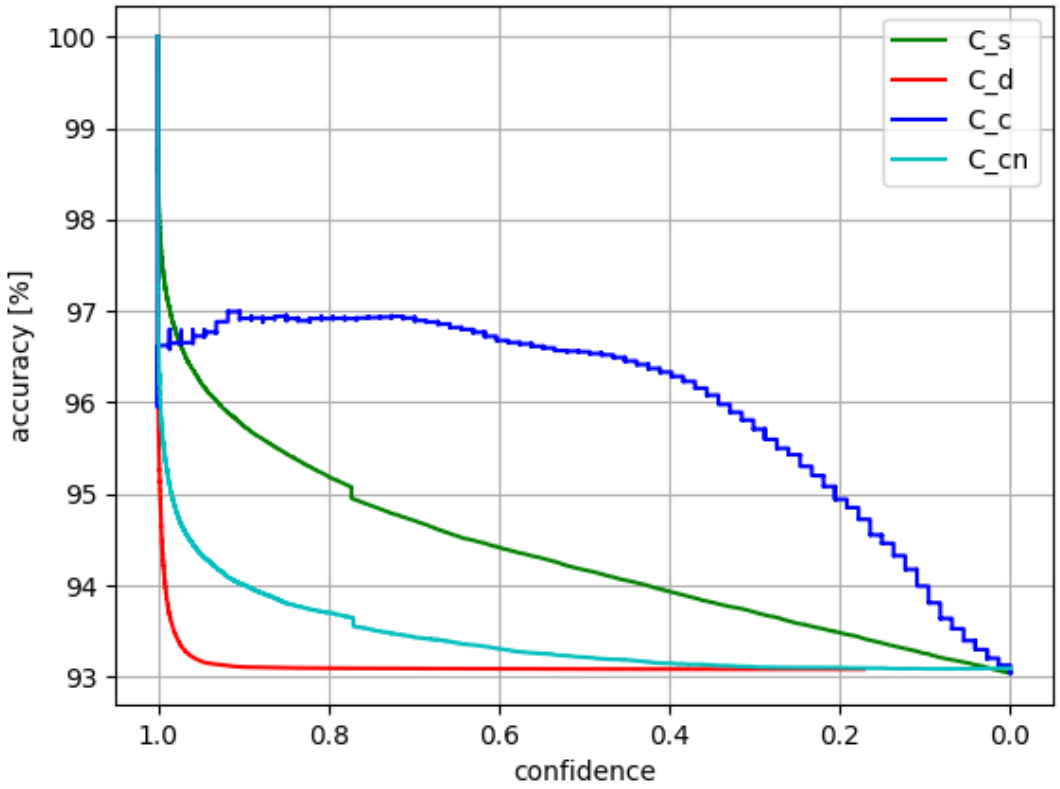

Figure 3.7. A plot showing the confidence-accuracy curve for shorter time series data using our robust model for target prediction. When preparing this plot, we order the outputs of our model based on confidence. The horizontal axis is the confidence, which was sorted in decreasing order. This plot does not have a proportional distribution of samples along the horizontal axis for most of the curves. The left side of the plot probably has a higher density of data. Accuracy on the vertical axis is the percentage of correctly classified samples of data, this accuracy is calculated in a cumulative way, so at each point in the plot the accuracy is calculated from current and all predeceasing data. C_s is the curve for the softmax output of the model, C_d is the curve for confidence acquired from test-time dropout, C_c is the curve for confidence calculated from the completeness of data (this curve has a proportional distribution of data along the horizontal axis) and C_{cn} is the curve for confidence acquired from the separately trained confidence classifier.

Chapter 4
Discussion and possible future work

4.1 Classifier of confidence

In the experiments we tested three confidences calculated during testing and an independent classifier for predicting the confidence of our robust model. We have came to a conclusion that the softmax response is the best of all of these providing a high quality selection/rejection option. We did not manage to train our independent confidence classifier to surpass or even get close to being as good as the softmax response. The reason for this may be that during training there was a big imbalance in the prior distribution of labels (our model had already high accuracy, so the label of 'wrong prediction' was rare). In theory it should be possible to train it to be better and more versatile than the softmax response.

Franc et al. in their paper propose more different approaches to this problem and a logical next step would be to try a different method, in an attempt to provide a higher quality selection/rejection option.

For our classifier of confidence we used the same architecture as for our target predictor, which was an adapted ResNet-50 model. A second approach to potentionally providing a better quality selection/rejection option is to try a different network architecture, which is proven to have the potential to increase the models effectiveness.

4.2 Change in data

A clear challenge is that a model trained in Austria (where TimeSen2Crop dataset [2] was sampled) would be deployed in different countries/continents changing the prior distribution and potentially could have crop types not seen before, which might confuse the model. A trained model fits the prior distributions and if they changed it would lower the accuracy. There are ways of recovering from the accuracy drop caused by a priors shift, which are disclosed in [8]. Moreover the entire data distribution may shift as well, which is known as the 'domain shift' and there are recent methods of 'domain adaptation' that allow updating the classifier in a self-supervised fashion or with minimal additional training. These are open-ended research questions in this direction.

4.3 Spatial recognition

Another limitation of the dataset [6] is that all the samples are for independent pixels. The narrow multispectral signature is surprisingly very discriminative. It would however help if something like a multispectral texture were represented. For example in the 'visible spectrum' computer vision it is difficult to distinguish, e. g. roads, buildings, vegetation, from RGB intensities of individual pixels. It would be much easier to do so from a larger area of the neighborhood. If we had the spatial localization of samples,

we could start solving problems like semantic segmentation. It would be then possible to find boundaries of the field, in addition to only recognizing crop type. This could be a topic of a next paper, building on the blocks of this thesis.

Chapter 5
Conclusion

To conclude our work, we resampled a publicly available dataset TimeSen2Crop [2] creating a dataset more suitable for the model. We resampled the data into shorter and also constant length. The data was then split into a training, validation and testing dataset. We designed a convolutional neural network to accept the data, by adapting the ResNet-50 model. We then trained two models of this network, by feeding them different inputs. We labeled them baseline model and robust model (which has a more unique concept). One was trained with complete time series and the other with shorter time series. We then compared them using the testing dataset, where the robust model was superior. We also tested multiple confidence scores, evaluating their quality using the standard coverage-accuracy curve. From our test we concluded that the best of them is the softmax response, which had an excellent accuracy to coverage trade off. Based on our tests we provided the robust model with a confidence score, which allowed the model to abstain from a decision. Using this method we significantly boosted the accuracy of the model traded for the coverage of decided samples i. e. the model only predicts for some samples.

References

[1] J. A. Richards. *Remote Sensing Digital Image Analysis: An Introduction. Berlin: Springer*. 2013.

[2] —. *"TimeSen2Crop: A million labeled samples dataset of sentinel 2 image time series for crop-type classification," IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 14, pp. 4699–4708*. 2021.

[3] Y. Geifman, and R. El-Yaniv. *"Selective classification for deep neural networks," in Proc. NIPS*. 2017.

[4] Y. Gal, and Z. Ghabramani. *"Dropout as a Bayesian approximation: representing model uncertainty in deep learning" in ICML,*. 2016.

[5] Vaclav Voracek Vojtech Franc, Daniel Prusa. *"Optimal strategies for reject option classifiers," Journal of Machine Learning Research,*. 2023, in Press..

[6] G. Weikmann, C. Paris, and L. Bruzzone. *"TimeSen2Crop dataset download"*. `https://zenodo.org/record/4715631`..

[7] Shaoqing Ren Kaiming He, Xiangyu Zhang, and Jian Sun. *"Deep Residual Learning for Image Recognition"*.
`arXiv:1512.03385v1`.

[8] M. Sulc, and J. Matas. *"Improving CNN classifiers by estimating test-time priors," in ICCV,*. 2019.