Crop Types Prediction with Temporal Convolutional Network

Yang Mu[a], Jingyan Li[b], and Yuru Jia[c]

[a]yangmu@ethz.ch
[b]jingyli@ethz.ch
[c]yurjia@ethz.ch

1. INTRODUCTION

The objective of this task is to implement and train a temporal convolutional network (TCN) that predicts crop types from optical satellite image (Sentinel-2) time series. Given pixel-wise RGB and NIR intensities with 71 temporal length, we are going to predict the pixel-wise crop labels at the next timestamp. We implemented a TCN with dilations and residual blocks, tested sample weights of loss function to handle the imbalance problem in the dataset, and searched best hyperparameters. Finally the best model achieved overall accuracy 81.91% on the test data set.

2. METHODOLOGY

2.1 Temporal Convolutional Network

A generic TCN consists of two principles: 1) the network produces an output of the same length as the input; 2)there is no information leakage from future to past. There are two general approaches to achieve these two points: 1) to use a 1D fully-convolutional network (FCN) architecture which ensures that each hidden layer is the same length as the input layer and thus the output layer is the same length as the input layer. 2) to use causal convolutions which only implements the convolution process based on the previous elements.

Dilated Causal Convolution. To apply the causal convolution on long time series, instead of using a simple causal convolution which can only look back at a history with size linear in the depth of the network, TCN uses a dilated causal convolution where the filter is applied over an area larger than its length by skipping input values with a certain step and allows the network to have very large receptive fields with just a few layers. Figure 1 shows the process of dilated causal convolution.

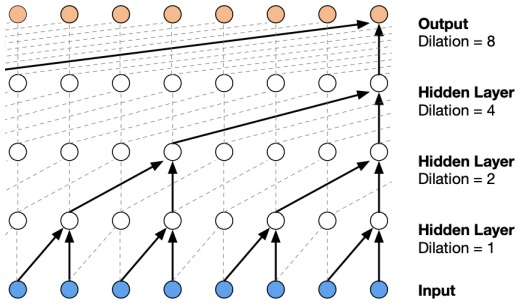

Figure 1. The dilated causal convolution with dilation factors d=1,2,4,8. Reference: WaveNet: A Generative Model for Raw Audio.[1]

TCN Residual Block. In order to stabilize the deep and large network, the residual block can also be used. As shown in the Figure 2, within a residual block, there are two layers where the dilated causal convolution is applied first, and followed by weight normalization and non-linear unit ReLU and additionally a spatial dropout. In terms of Residual block, instead of adding the output to the input directly, an additional 1×1 convolution is used to ensure the element-wise addition since there could have difference width between the input and the output.

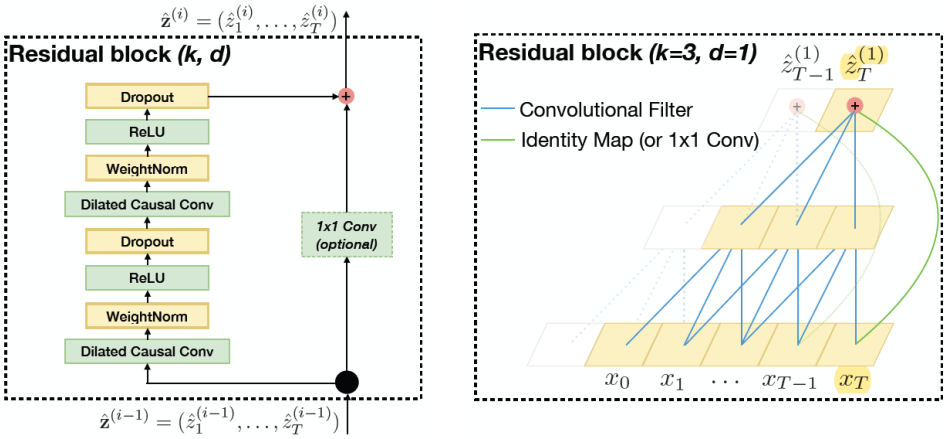

Figure 2. TCN residual block. Reference: An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling.[2]

2.2 Adaptation to multi-class classification

In the lab, we are supposed to conduct multi-class classification to predict the crop type at next timestamp. The output length of TCN in our scenario is then set as one. Therefore, we only extracted last element in the output of TCN and connected it to a fully connected layer. Then we applied a softmax layer of 13 channels (i.e. the number of crop types in the data set) after the fully connected layer to get probabilities per crop type.

2.3 Sample Weights for imbalanced classes

In our task, the distribution of classes are imbalanced. When training a model on an imbalanced data set, the learned model becomes biased towards the classes with majority samples. Since the majority classes offer more samples during training, the model learns to perform well on the majority classes. On the other side, due to the lack of enough samples, the model fails to learn meaningful patterns over minority classes. So it is possible that our model cannot perform in minority classes as well as in majority classes.

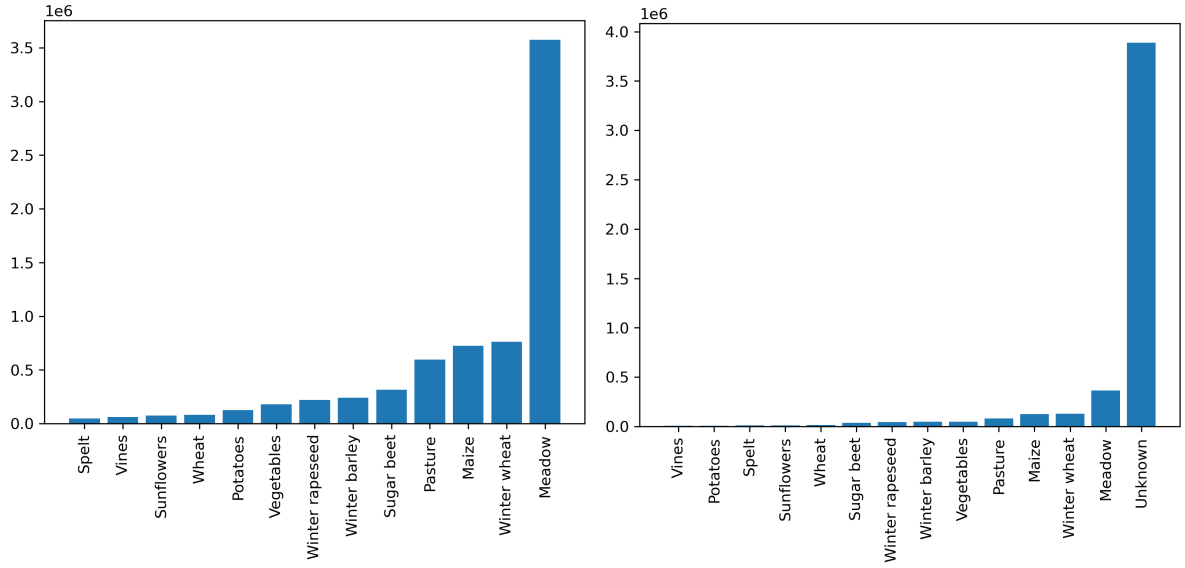

Figure 3. (Left): Histogram of different classes in train data set. (Right): Histogram of different classes in test data set.

To solve the imbalance, we utilized sample weights in the loss function during training. The idea is to weigh the loss computed for different samples differently based on the class they belong to. Intuitively, we assign a

higher weight to the loss encountered by the samples in minor classes. We used Class-Balanced Loss Based on Effective Number of Samples[3] (ENS) as the sample weight in loss function. As the number of samples increases, the additional benefit of a newly added data point will diminish. ENS then utilizes the effective number of samples of a class to define the weight. According to (1), for a class c with n samples, the weight $w_{n,c}$ is defined as the inverse of the number of effective samples E_{n_c}. The effective number of samples is defined as the volume of samples. n is the number of samples and $\beta \in [0, 1)$ is a hyperparameter. As suggested by the authors, β can be selected from 0.9, 0.99, 0.999, 0.9999.

$$w_{n,c} = \frac{1}{E_{n_c}}, E_{n_c} = \frac{1-\beta^{n_c}}{1-\beta} \tag{1}$$

3. EXPERIMENTS

We trained TCN of different versions. The training data was split into 80% for training and 20% for validation. The models were then evaluated on test dataset by overall accuracy, precision, recall and f1 score per crop type, as well as the visualization map.

3.1 Hyperparameter Tuning

In the hyperparameter tuning, we used an open source framework Optuna to search hyperparameter automatically. Optuna excels at using the pruning algorithm during the optimization, and it supports automatically stopping unpromising trials (i.e., combinations of hyperparameters) at the early stages of the training. Another advantage is that it provides the quick visualization for hyperparameter optimization analysis, including the optimization history, parameter importances and high-dimensional parameter relationships, etc.

We selected *levels* (number of hidden layers), *n_hunits* (number of hidden units per layer), *lr* (learning rate), *kernel_size* and *dropout* as the hyperparameters to be tuned. Due to the time limitation, 30% of data was randomly sampled and split to train and validation set, on which we have run 15 trails to find the best hyperparameters to get the highest validation accuracy.

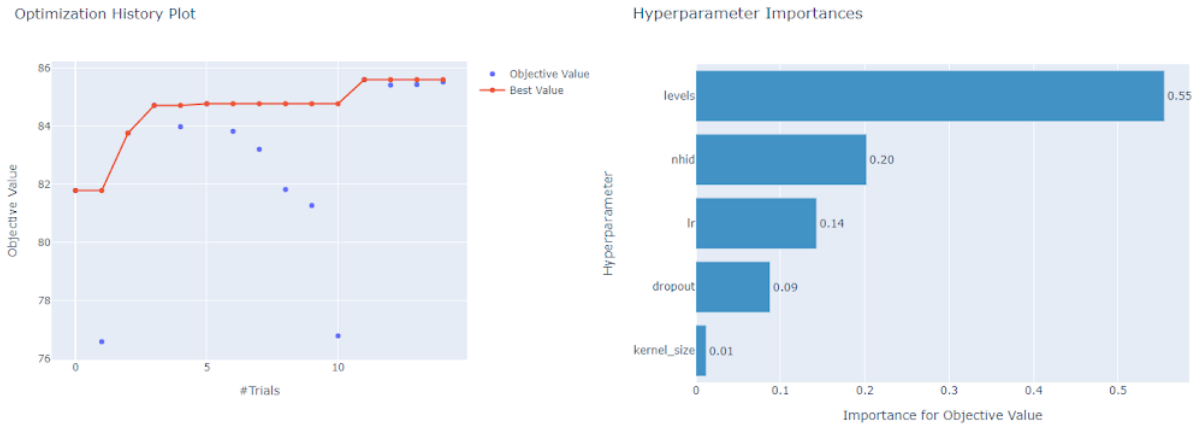

Figure 4. (Left): Optimization history (Right): Parameter importances

The optimization history is shown in Figure 4 (Left), we can see that at the trail 11 the best value of objective has been obtained. The importance of parameters is shown in Figure 4 (Right), the levels and number of hidden units show higher importance compared to other hyperparameters. The final parameters we got is shown in Table 1.

Table 1. Best hyperparameters

lr	dropout	n_hunits	kernel_size	level
4e-4	0.01	40	5	6

3.2 Using Residual Blocks

We compared the prediction results of models either using residual blocks or not. The overall accuracy of simple TCN on test set is 81.91%. The metrics are shown in Figure 5 (Left). TCN with residual blocks is shown in Figure 5 (Right). The overall accuracy is 80.97%.

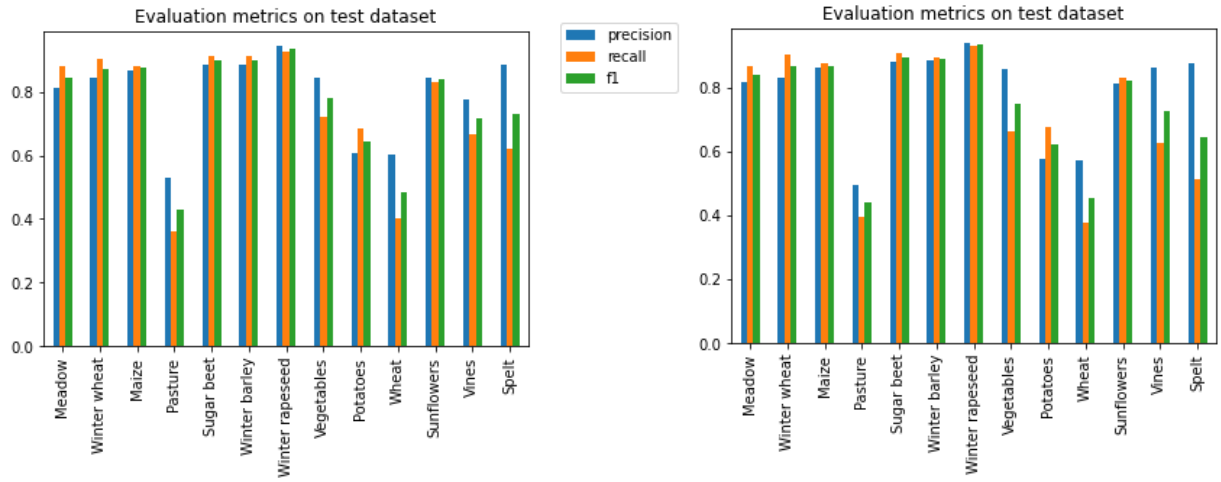

Figure 5. (Left): The model is trained by simple TCN. (Right): The model is trained by TCN with residual blocks. The evaluation metrics over test data set. From right to left, class names are sorted according to the number of samples in the train data set.

3.3 Adding Weights in Loss Function

We compared the prediction results either using sample weights or not during training. We set β to be 0.99 in the case because the value showed better overall accuracy than the other recommended alternatives. The overall accuracy is 81.17%. As shown in Figure 6, the precision of wheat and spelt has been greatly improved after adding weights (approx. 5%), the remaining categories have not changed much. It improves the prediction accuracy on minority labels, since wheat and spelt are the types of minimum samples in our dataset.

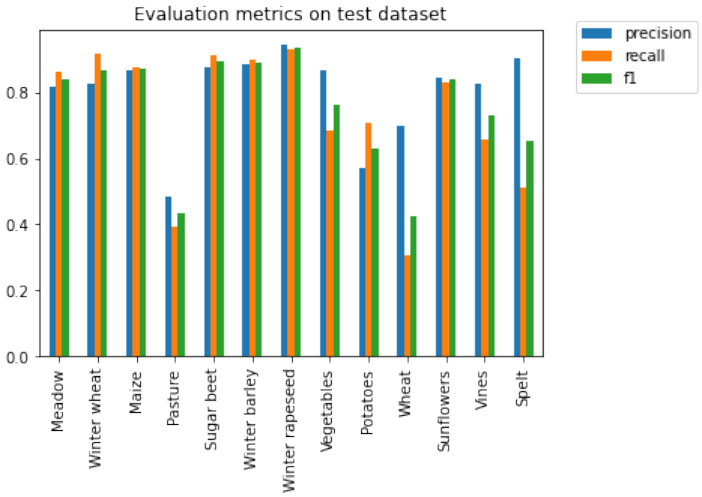

Figure 6. The model is trained with ENS sample weights. The evaluation metrics over test data set. From right to left, class names are sorted according to the number of samples in the train data set.

4. RESULT

By comparing the results for different experiments, we find that simeple TCN achived the highest accuracy of 81.91% on the test dataset, and the corresponding consuming time and memory requirements are shown in 2. We visualized the prediction of Simple TCN model, and the ground truth of the test data and the prediction are shown in Figure 7 and Figure 8 respectively.

Table 2. Overall accuracy, time consuming and memory requirements result for the model.

Overall accuracy	Training time	Inference time	Memory requirements
81.91%	41640s	660s	4GB(GPU)

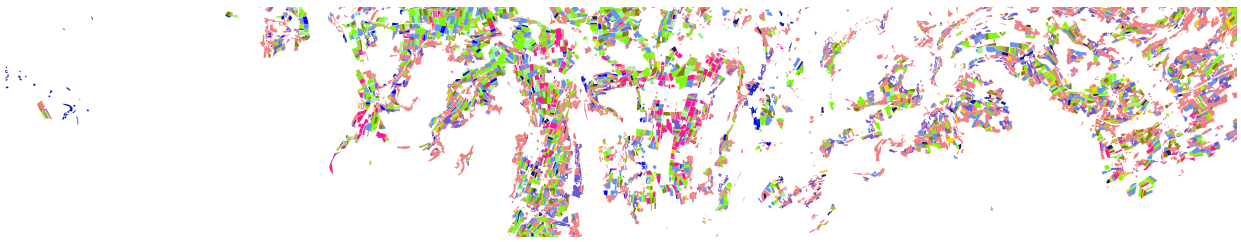

Figure 7. The ground truth of the test data.

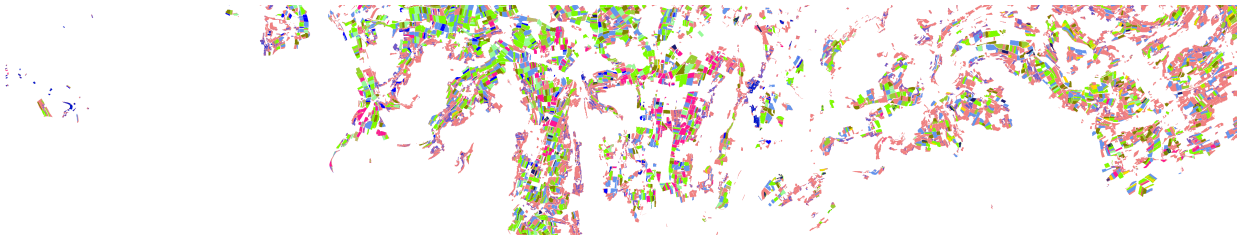

Figure 8. The prediction of the test data.

In order to evaluate the accuracy of the prediction more clearly, we indicate pixels with correct predictions in green and wrong predictions in red, and Figure 9 shows the generated true/false map. It can be seen that most of pixels are predicted correctly, and only a few misclassified pixels are distributed throughout this map.

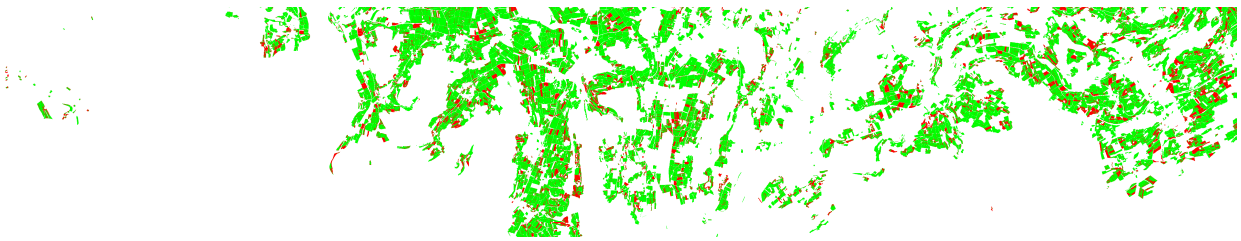

Figure 9. True/False map. Green indicates the correct prediction and red indicates wrong prediction.

A small window was chosen to analyze the prediction into details as shown in Figure 10. Such as the area highlighted by the red rectangle, the main classification errors are the crops represented by red (Meadow) and blue (Pasture). In addition, the ground truth is constant for a patch of crop land, however in our prediction result, some misclassifications appear as small dots in a patch of crop land, just like the area framed by the blue rectangle. Pasture shows similar reflection characteristics to Meadow, thus it is usually confused with Meadow in our prediction.

The confusion matrix is shown in Figure 11, from which we can see that the most confusing crops to classify are meadow and pasture, this may because they both belong to green vegetation and are very similar in property.

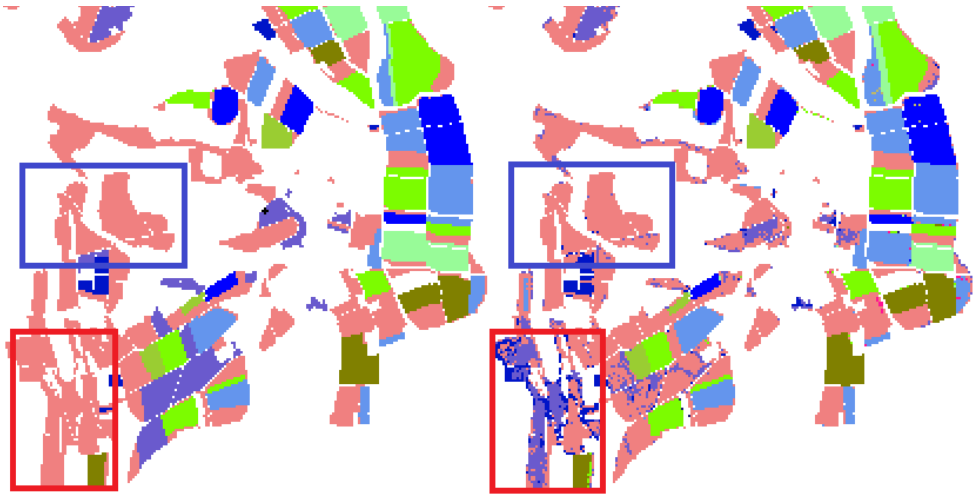

Figure 10. (Left): Small window of ground truth (Right): Small window of prediction.

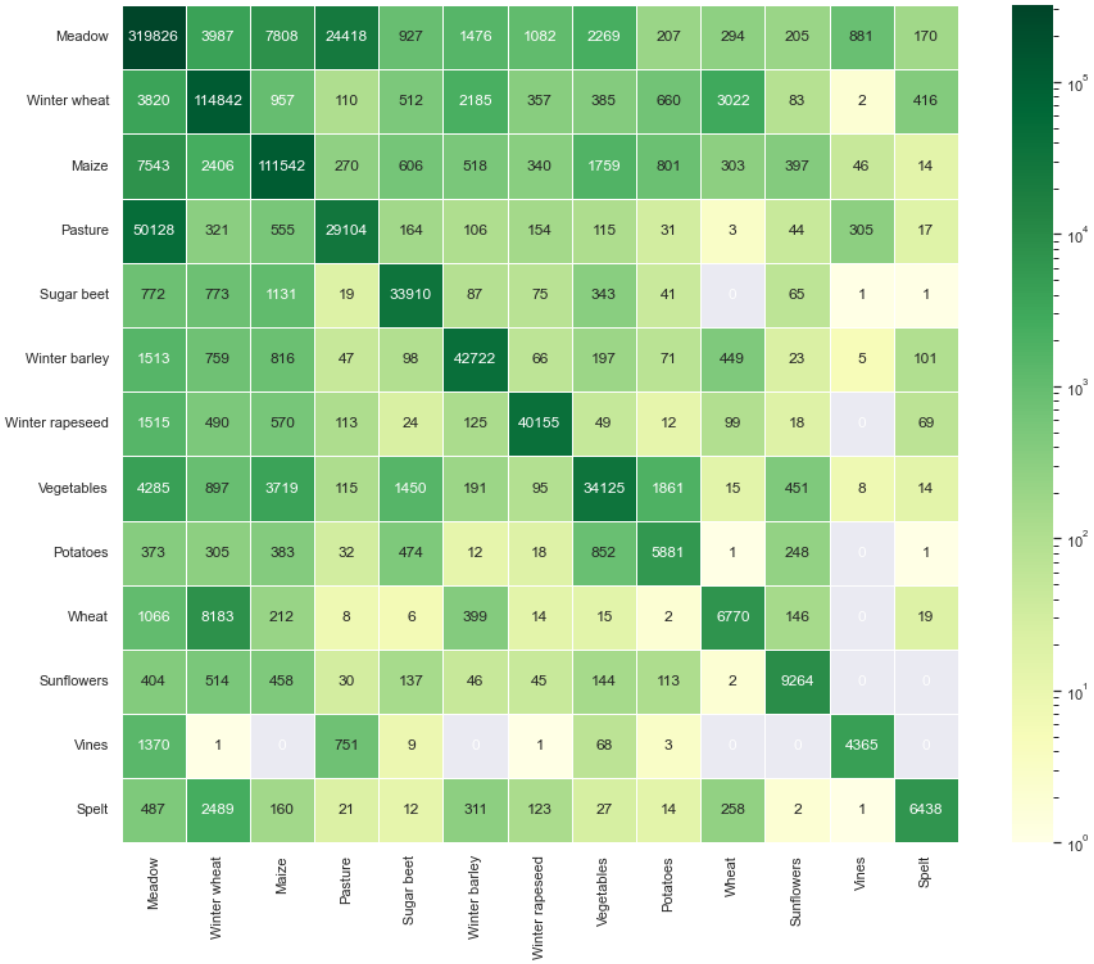

Figure 11. Confusion matrix of the prediction

The precision, recall and f1 score of each crop type are shown in the Table 3. The winter rapeseed got the highest three scores while the pasture got the lowest. Overall, the simple TCN model made a good prediction.

Table 3. Performance comparison w.r.t. crop types.

Crop types	Precision(%)	Recall(%)	F1(%)
Meadow	81.4	88.0	84.5
Winter wheat	84.5	90.2	87.2
Maize	86.9	88.1	87.5
Pasture	52.9	35.9	42.8
Sugar beet	88.5	91.1	89.8
Winter barley	88.7	91.2	89.9
Winter rapeseed	94.4	92.9	93.6
Vegetables	84.6	72.3	77.9
Potatoes	60.6	68.5	64.4
Wheat	60.4	40.2	48.3
Sunflower	84.6	83.0	83.8
Vines	77.8	66.5	71.7
Spelt	88.7	62.2	73.1

5. DISCUSSION AND CONCLUSION

During the experiment, we firstly trained the simple TCN model to do the prediction, and used the open source framework Optuna to optimize the hyperparameters, finally got an overall test accuracy of 81.91%.

Then we found an imbalance class distribution in the data, so we tried to add ENS weights during the training, which has improved the precision of wheat and spelt greatly, however, the overall test accuracy is slightly lower than the simple TCN model. In addition, we tried the TCN model with residual blocks, unfortunately it was still sightly lower than the results of simple TCN model. We guess the main reason is that the best hyperparameters found on the simple TCN model were used in all experiments, due to the time limit, we didn't try to find the best hyperparameters on TCN with ENS weights and with residual blocks.

Regarding the prediction result, our model predicts sugar beet, winter barley, winter rapeseed and spelt quite well (i.e. precision 88%), but cannot well predict pasture, potatoes and wheat (i.e. precision $< 70\%$). Besides, some crops with similar reflection characteristics cannot be separated well if we observe the confusion matrix in Figure 11. For example, the model confuses in predicting either meadow or pasture, wheat or winter wheat.

To conclude, all of these TCN model made good predictions, with the overall test accuracy exceeded than 80%, with the best predicted crop category being winter rapeseed and the worst being pasture.

REFERENCES

[1] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K., "Wavenet: A generative model for raw audio," *CoRR* **abs/1609.03499** (2016).

[2] Bai, S., Kolter, J. Z., and Koltun, V., "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *CoRR* **abs/1803.01271** (2018).

[3] Cui, Y., Jia, M., Lin, T.-Y., Song, Y., and Belongie, S., "Class-balanced loss based on effective number of samples," in [*Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*], 9268–9277 (2019).