Assignment 2 Report: Time Series Analysis

Alize Sevgi Yalçınkaya
34790

May 18, 2024

Introduction

Importing Data

The Breizhcrops datasets, when imported, are unique datatypes. All 4 datasets corresponding to a region were imported. Because the UNIXTIME within the time series sequences was going to be useful, they were imported with raw_transform. Similarly, bottom of atmosphere level (L2A) was used to ensure ease of use and better results.

Data Preprocessing

All 4 datasets were converted to dataframes and saved as pickle files because they are quite large. This was done using the dataframe_creation function and aided by simplify_structure. Because of the scope of the datasets and personal limited resources, most of the outputs are absent from the provided Jupyter notebook because they were run separately, on different resources, or on remote servers. Because the dataframes (and later numpy arrays) are saved to the directory, these operations were able to be done in a modular fashion. In the paper, deep learning models used a sequence length of 45 achieved by subsampling, so that is also what is done in this project [1]. But since L2A level observations contain shorter sequences than L1C level data, interpolation was also used to increase the sequence lengths of those that were shorter than 45.

1. Subsampling: If sequence length is longer than 45, we subsample to 45.
2. Interpolation: If sequence length is shorter than 45, we use interpolation in a way that doesn't mess with our original data points, so interpolates only at new times.
3. Create features: We take only the time_series column, drop the UNIXTIME, save as .npy files.
4. One hot encode labels: 9 classes one-hot-encoded and saved as .npy files.

1 Part 1: Baseline Architecture

Dictionaries were used to import the correct training and test sets for different parts.

1.1 LSTM Model

Since the aim of this assignment is domain generalization (and due to a lack of resources, in particular, a GPU) a simpler LSTM model was opted for. The model has 3 LSTM layers followed by Dropout

Fold	Train Data	Test Data
1	frh01, frh02, frh03	frh04
2	frh01, frh02, frh04	frh03
3	frh01, frh03, frh04	frh02
4	frh02, frh03, frh04	frh01

Table 1: Train and Test Data for Each Fold

layers and a single Dense layer. Adam optimizer and categorical_crossentropy were used. The hyperparameters were not tuned but were inspired by the preexisting LSTM model present in the Breizhcrops GitHub [?]ue to the same resource limitations.

1.2 Metrics

The main metrics used in the paper [2] were overall accuracy, average accuracy, weighted F-score, and the kappa metric.

Test	Overall Accuracy	Average Accuracy	Weighted F-score	Kappa Metric
frh04	0.811357	0.958079	0.800956	0.751689
frh03	0.801926	0.955984	0.800150	0.741494
frh02	0.803256	0.956279	0.797245	0.743776
frh01	0.829364	0.962081	0.830086	0.783737

Table 2: Performance Metrics for Different Test Sets

2 Part 2: Lower and Upper Performances

First, x was chosen as 80 per the assignment definition but that did not yield a lower lower-baseline than upper-baseline so x was changed to 60. The performance is still not great because the 4th region (1st fold) in lower baseline still outperforms the upper baseline but for the other folds, this is not a problem even though the differences are minute.

Test	Overall Accuracy	Average Accuracy	Weighted F-score	Kappa Metric
frh04_60	0.813053	0.958456	0.805246	0.754561
frh03_60	0.802056	0.956012	0.789205	0.739132
frh02_60	0.797786	0.955064	0.788374	0.736217
frh01_60	0.830379	0.962306	0.831136	0.784749

Table 3: Performance Metrics for Lower Baseline

Test	Overall Accuracy	Average Accuracy	Weighted F-score	Kappa Metric
frh04_60	0.808829	0.957517	0.801871	0.749127
frh03_60	0.813005	0.958446	0.807170	0.754602
frh02_60	0.811495	0.958110	0.808970	0.755063
frh01_60	0.839018	0.964226	0.837874	0.794876

Table 4: Performance Metrics for Upper Baseline

3 Part 3: Domain Generalization

Aligning the features extracted from different regions means making sure that the features learned from the three training domains are mapped to a similar space. This helps the model generalize better to a new, unseen domain by reducing the domain shift.
In Domain-Adversarial Neural Networks (DANN), this is done by the adversarial training process where the feature extractor tries to fool a domain classifier, and the domain classifier tries to distinguish between domains. This forces the features from different domains to be aligned. [3]

- **Feature Extractor**: The LSTM layers act as the feature extractor.
- **Gradient Reversal Layer**: The GradientReversal layer inverts the gradients during backpropagation, encouraging the feature extractor to produce domain-invariant features.
- **Domain Classifier**: The domain_preds head tries to predict the domain of the features. The adversarial training ensures that the feature extractor learns to produce features that the domain classifier cannot easily distinguish, aligning the features from different domains.

For our task, the domain classifier was not able to predict a single test data correctly because in our training data, we only used 3 regions. But, we were able to visualize the feature overlap (alignment) using PCA. The code combines features from all folds and plots them using PCA, colour-coded by domain labels to visualize feature alignment.[4]

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 45, 13)]	0	[]
lstm (LSTM)	(None, 45, 128)	72704	['input_1[0][0]']
dropout (Dropout)	(None, 45, 128)	0	['lstm[0][0]']
lstm_1 (LSTM)	(None, 45, 128)	131584	['dropout[0][0]']
dropout_1 (Dropout)	(None, 45, 128)	0	['lstm_1[0][0]']
lstm_2 (LSTM)	(None, 128)	131584	['dropout_1[0][0]']
dropout_2 (Dropout)	(None, 128)	0	['lstm_2[0][0]']
gradient_reversal (GradientReversal)	(None, 128)	0	['dropout_2[0][0]']
dense (Dense)	(None, 128)	16512	['gradient_reversal[0][0]']
dropout_3 (Dropout)	(None, 128)	0	['dense[0][0]']
class_preds (Dense)	(None, 9)	1161	['dropout_2[0][0]']
domain_preds (Dense)	(None, 4)	516	['dropout_3[0][0]']
Total params:			354061 (1.35 MB)
Trainable params:			354061 (1.35 MB)
Non-trainable params:			0 (0.00 Byte)

Table 5: Summary of LSTM with DANN for a fold

Test	Overall Accuracy	Average Accuracy	Weighted F-score	Kappa Metric
frh04_60	0.813929	0.958651	0.808071	0.755615
frh03_60	0.804236	0.956497	0.804014	0.745159
frh02_60	0.788983	0.953107	0.774427	0.724648
frh01_60	0.833744	0.963054	0.834084	0.788972

Table 6: Performance Metrics for DANN Results

Test	Domain Accuracy
frh04_60	0.0
frh03_60	0.0
frh02_60	0.0
frh01_60	0.0

Table 7: Domain Classification Accuracy

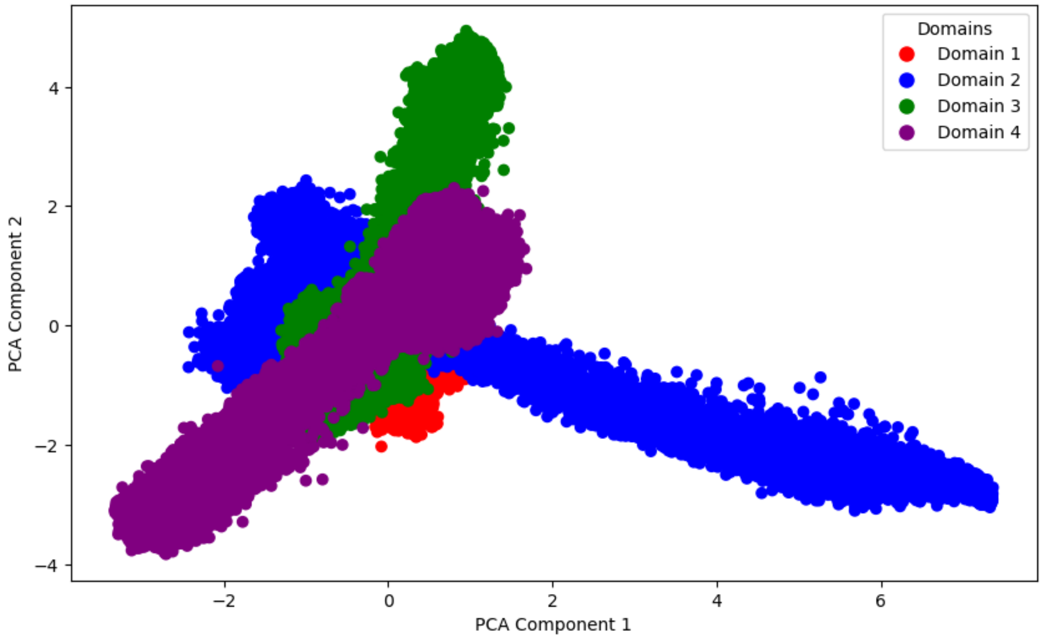

Figure 1: PCA plot showing feature alignment across domains using DANN model

Conclusions

- The DANN model shows effective feature alignment across domains, as evidenced by the PCA plot and consistent performance metrics.
- The improvement is relatively consistent across various regions, indicating that the DANN model can generalize well to unseen domains. The overall accuracy, average accuracy, weighted F-score, and kappa metric are relatively consistent across different folds. The exact performance may vary slightly depending on the region left out, but overall, the model maintains good generalization.
- The overlap indicates successful domain adaptation, suggesting that the features extracted from the three training regions generalize well to the test region.
- The distinct clusters hint at areas where feature alignment could be further improved, especially for domains with less overlap.

References

[1] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[2] Marc Rußwurm, Charlotte Pelletier, Maximilian Zollner, Sébastien Lefèvre, and Marco Körner. Breizhcrops: A time series dataset for crop type mapping. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences ISPRS (2020)*, 2020.

[3] Pumpikano. tf-dann: Tensorflow implementation of domain-adversarial neural networks, 2019. Accessed: 2024-05-16.

[4] Packt Publishing. Hands-on computer vision with tensorflow 2: Chapter 7 - train a simple domain adversarial network (dann), 2019. Accessed: 2024-05-15.

[5] Talip Ucar. Pytorch implementation for Domain Adaptation, Alignment and Translation. `https://github.com/talipucar/PyFlow_DomainAdaptation`, since 2021.

[6] Marc Rußwurm, Charlotte Pelletier, et al. Breizhcrops: A time series dataset for crop type mapping, 2023. Accessed: 2024-05-09.

[7] Vivien Sainte Fare Garnot, Loic Landrieu, Sebastien Giordano, and Nesrine Chehata. Satellite image time series classification with pixel-set encoders and temporal self-attention. *CVPR*, 2020.

[8] Slobodan Sjelic. vojvodina_crop_classification: Vojvodina crop classification using sentinel-2 data, 2020. Accessed: 2024-05-11.

[9] Jindong Wang et al. Everything about transfer learning and domain adapation. `http://transferlearning.xyz`.

[10] Wei-Chih Kuo, Yi-Lun Tsai, Winston Hsu, and Shu-Pin Hu. A coarse-to-fine indoor layout estimation (cfile) method based on domain transfer and line segments. *arXiv preprint arXiv:2103.03097*, 2021.