Applied Deep Learning, Exercise 3: Generating global climate maps by clustering Vegetation Optical Depth encodings

Leander Moesinger, ID:11937136

January 20, 2021

1 Introduction

Current global climate classifications often rely on some hand-crafted rules based on topic knowledge. E.g the Koeppen-Geiger classification (fig. 1) segments the Earth into different climates based on its creators experience as a botanist and is based on logical statements using precipitation and temperature. From a practical standpoint, there probably was a lot manual tweaking involved in finding the optimal parameters to generate a map that matched the expected output. Also, the number of classes is static, therefore an user can not segment the globe into finer climate classes if desired.
This process is automatized using deep learning: Vegetation, precipitation and temperature data are encoded using an auto-encoder and the encoding then used to cluster the globe. The application is a small program where an user can enter a number and receives a global map segmented into that many climate classes.

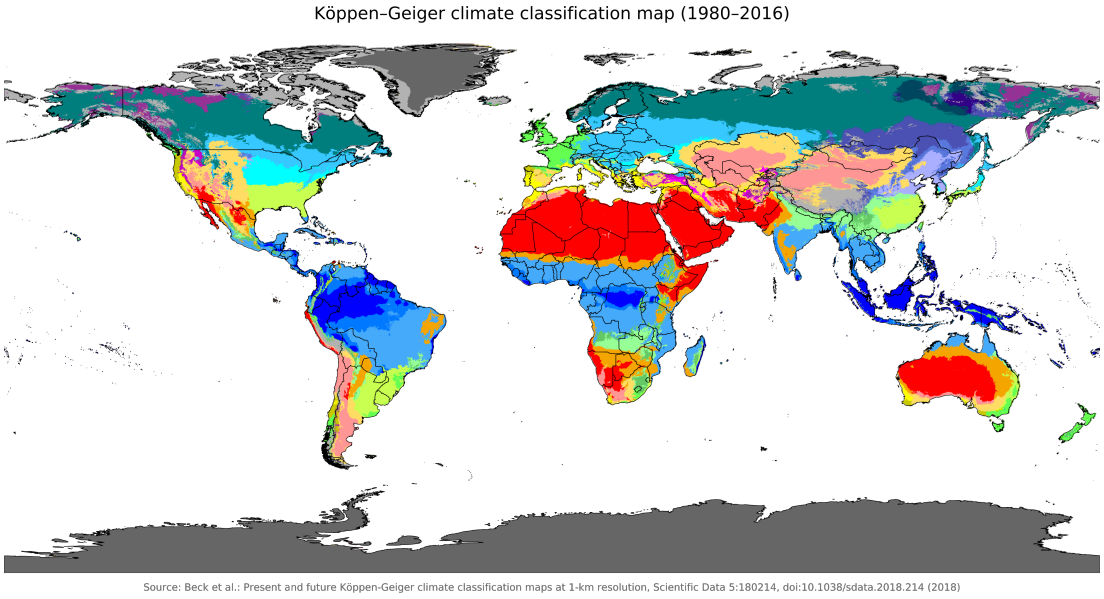

Figure 1: Up to date Koeppen-geiger classification (Beck et al., 2018).

2 Data & preprocessing

2.1 VOD

The Earth radiates microwave radiation. Part of this radiation is attenuated by water in vegetation. This degree of attenuation is described by Vegetation Optical Depth (VOD). Fig 2 shows the global mean VOD from 2002-2017. If VOD is high, such as in the tropics or Boreal forests, it means that there is a lot of vegetation or that the vegetation is very wet. If VOD is low, such as in the deserts, it is because the vegetation is dry or because there is no vegetation in the first place. VOD is also dynamic over time (fig 3) due to vegetation growth and the seasonal cycle. VOD is unitless and theoretically ranges from 0 (no attenuation) to infinity (all radiation is attenuated), but practically ranges from 0 to about 1. VOD contains a lot of information about the vegetation, and therefore it should be useful to generate a climate classification from it.

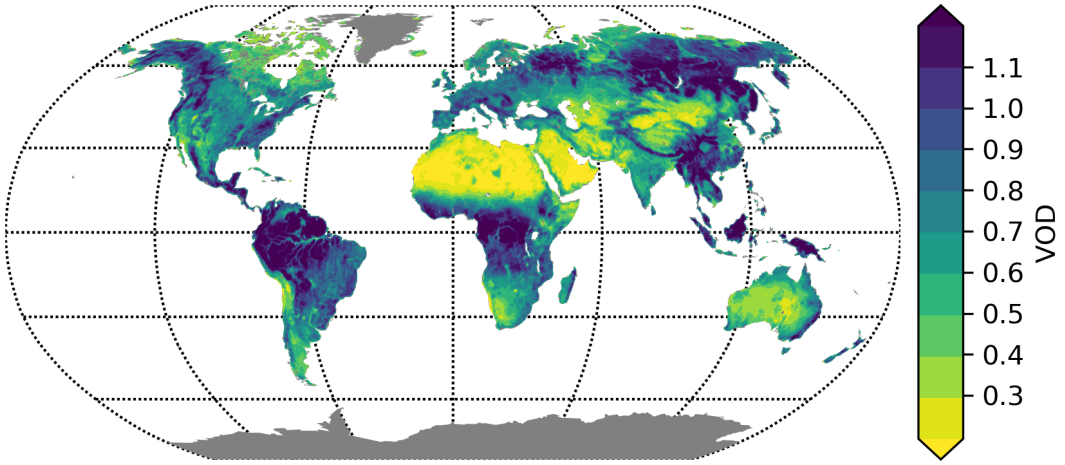

Figure 2: Mean VOD between 2002 and 2017 (Moesinger et al., 2020).

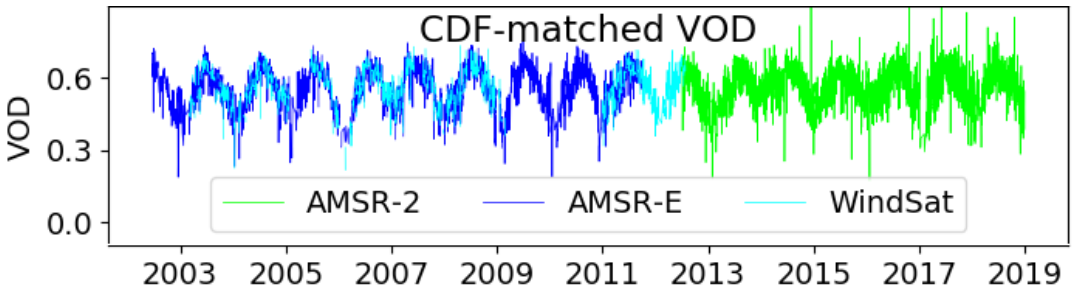

Figure 3: VOD time series measured by various spaceborne sensors for a location in Austria. It is high in summer, low in winter (Moesinger et al., 2020).

We are going to use VOD distributed via the VOD Climate Arcive (VODCA), an open-access[1] VOD database with global extent, that stretches over the past 30 years. It has a quarter degree spatial resolution and a daily temporal resolution (Moesinger et al., 2020).

The VOD-data are downsampled to weekly means. There are a few reasons for downsampling:

[1] https://zenodo.org/record/2575599

- The data has missing values, and by taking weekly means we reduce the number of gaps.
- The original dataset is quite large, downsampled (and by dropping some unnecessary columns) it is at 13.3GB.
- The original data is quite noisy and sub-weekly variations are more a result of noise rather than vegetation changes.

Also, the the time series on the southern hemisphere were shifted by 6 months so the seasons align with the northern seasons. Finally, the data are standardized before feeding it to the network to make them use the whole possible range and centered around 0 and missing values were replaced with 0.

2.2 Precipitation and temperature data

The mean precipitation and temperature from ERA5 (Hersbach et al., 2020) are used, which is a global land surface model using various data sources. We take the temporal means (over the whole time period, resulting in a single global image), which are side tasks for the autoencoder to predict from the encoding. The data were also standardized.

3 Methods

The general setup can be seen in fig. 4. At the core we have an autoencoder that takes VOD data, encodes it, and decodes it back to VOD. As side tasks, starting from the encoding precipitation and temperature are also predicted using some small networks. Once the encoding is trained, the encoding is given to a k-means clusterer to generate the climate classes.

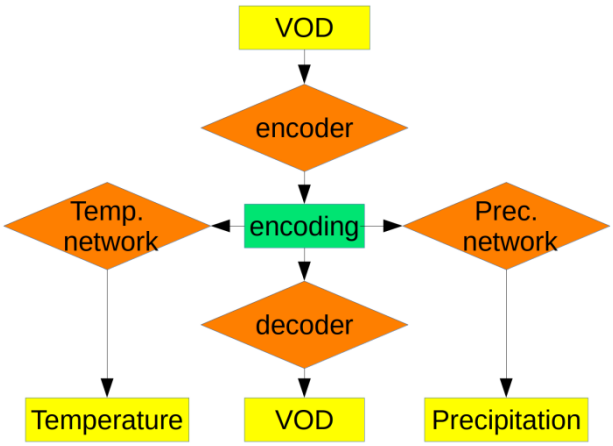

Figure 4: Overview of the neural network. The center column (VOD to VOD) is an autoencoder, the two side networks are two simple feedforward networks solving the side tasks

3.1 Autoencoder architecture

An autoencoder is used to extract features and cluster them with a shallow learner. An autoencoder (fig. 5) is an encoder/decoder setup that takes a VOD time series as input, encodes it into some low dimensional latent representation, and then tries to reconstruct the original time series again.

After trying out various setups, one that is very similar to Richard et al. (2020) proved to give the best results. The autoencoder uses convolutions to detect patterns and a symmetric encoder/decoder setup. In detail, the encoder starts with a dropout layer to simulate more missing data to prevent the autoencoder from learning "missing data" patterns in the time series. It is then followed by two convolution layers and

two dense layers. The second dense layer has an output size of 4, which is the size of the encoding. The decoder uses a mirrored setup, with first two dense layers followed by two deconvolution layers. Batch-normalization is done after each layer to speed up learning, and activation function is elu, except before the encoding where it is sigmoid (to force the encoding to be in [0, 1], which is easy to interpret and plot) and the last layer of the decoder which is linear (to allow it to reach values from -inf to inf).

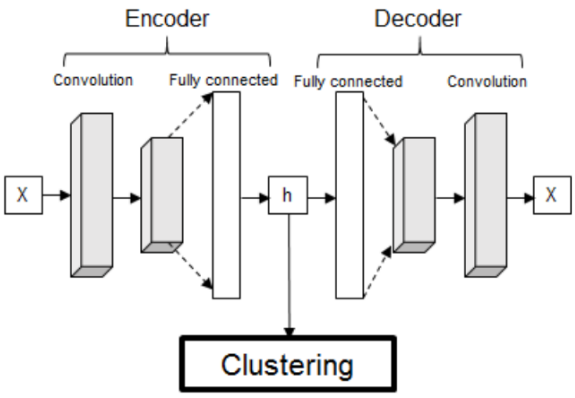

Figure 5: Overview of used autoencoder setup (Richard et al., 2020).

3.2 Side task networks

The networks to predict temperature and precipitation are simple and small feed-forward neworks consisting of one dense layer with 8 neurons (relu activation), and a second dense layer with one neuron (linear activation). The purpose of this side task is to force the network to train an encoding that is actually useful and contains information about the climate and is not just a compression of the data.

3.3 Error metric and missing data handling

As loss the mean squared error of the VOD reconstruction error, the temperature prediction and the precipitation prediction were added together. What is important here is the handling of missing data: While the network tries to predict input nans, they are masked out before doing the error calculation. Therefore it wont get penalized for not predicting where a gap in the data is.

3.4 Application and clustering

The application is a simple python script that can be called via command line. It takes as arguments 1) the path to an encoding 2) the number of clusters to generate and optionally c) the tolerance of the kmeans (indicating when to it should stop optimizing). The script then reads the encoding, clusters it with kmeans and saves them as a netcdf4. Additionally, it also produces two .png images, which contain plots of the created clusters and colored according to the first three principal components of the encoding using them as RGB values (e.g. fig. 6). Clusters with a similar mean encoding will therefore end up with similar colors.

4 Results

Fig. 6 shows an example output, where the encodings were grouped into 20 clusters. The network is not using any spatial information, yet it still produced spatially mostly coherent clusters. The clusters generally make sense, there are no sudden jumpy in color between neighbouring clusters. It looks roughly similar to the Koeppen-geiger map (fig. 1), where e.g. violet corresponds to the equatorial climate or red

to the hot arid climate. The biggest differences can be found in the snow climate, where the autogenerated clusters have a very different pattern. However, i am not enough an expert in climate classes to tell whether the autogenerated clusters are nonsense.

Another benefit is the encoding itself, which is only 2.8 mb large, compared to the $\sim$ 13 GB of weekly VOD data alone. It could be added as a feature to a number of problems, such as fire hazard monitoring or as input in a land surface model.

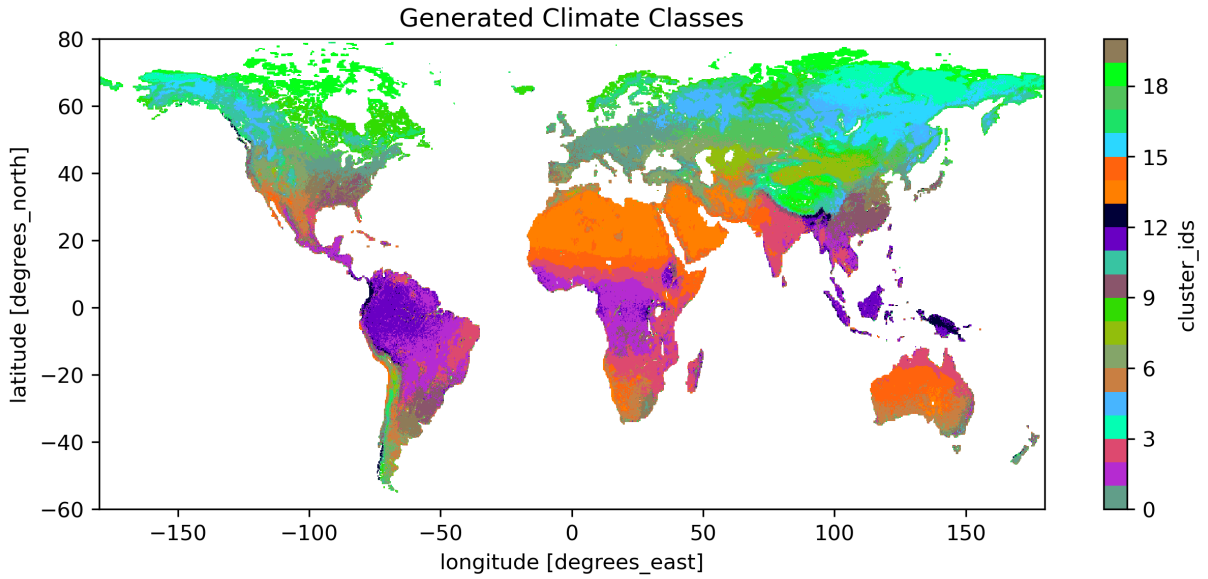

Figure 6: Example output with 20 classes. Similar colored clusters indicate that they have a similar encoding

5 Lessons learned

During the project my main take-away was to not spend too much time tweaking hyperparameters and adding or removing a layer. It likely wont make that big of a difference, as the networks are powerful enough to compensate a sub-optimal setup. But what really makes a difference is the input data and preprocessing. E.g. adding the side-tasks improved the result drastically, or shifting the values on the southern hemisphere by 6 months (before the network had a tendency to also learn whether a time series is on the south or north hemisphere, which is not useful for a climate classification). Also, what really made the tweaking of hyperparameters time consuming is randomness; You can run the same network twice and get quite different outputs. In the future, if i would do this again, i would set my code up in a way that it always trains 10 networks in one go and only saves the best one to get a bit a more reliable results.

Specific to autoencoders, i found that one has to be careful that they do not learn the wrong thing, such as missing data patterns or whether a time series is on the north or south hemisphere. Having a good reconstruction error alone does not guarantee that the encoding is actually useful.

6 Time spent

I lost a bit track of time, but i spend a lot more time than estimated on fine tuning networks than planned, it took more time than everything else combined and also did not improve the results drastically. On

the other hand, the application took a lot less time, but mostly because i scrapped it in extent. In the beginning i planned to make some nice GUI for it and ship it in a docker container, but i ran out of time and have now a simple script that can be executed with some args. The video also took a lot more time than expected - its amazing how i freeze up in front of a camera, i had to record every sentence countless times. In total i think i spent something between 120-160 hours, which is a bit above my estimated time in the beginning.

References

Beck, H. E., Zimmermann, N. E., McVicar, T. R., Vergopolan, N., Berg, A., and Wood, E. F. (2018). Present and future köppen-geiger climate classification maps at 1-km resolution. *Scientific Data*, 5.

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., Simmons, A., Soci, C., Abdalla, S., Abellan, X., Balsamo, G., Bechtold, P., Biavati, G., Bidlot, J., Bonavita, M., De Chiara, G., Dahlgren, P., Dee, D., Diamantakis, M., Dragani, R., Flemming, J., Forbes, R., Fuentes, M., Geer, A., Haimberger, L., Healy, S., Hogan, R. J., Hólm, E., Janisková, M., Keeley, S., Laloyaux, P., Lopez, P., Lupu, C., Radnoti, G., de Rosnay, P., Rozum, I., Vamborg, F., Villaume, S., and Thépaut, J. N. (2020). The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049.

Moesinger, L., Dorigo, W., de Jeu, R., van der Schalie, R., Scanlon, T., Teubner, I., and Forkel, M. (2020). The global long-term microwave vegetation optical depth climate archive (vodca). *Earth System Science Data*, 12(1):177–196.

Richard, G., Grossin, B., Germaine, G., Hébrail, G., and de Moliner, A. (2020). Autoencoder-based time series clustering with energy applications.