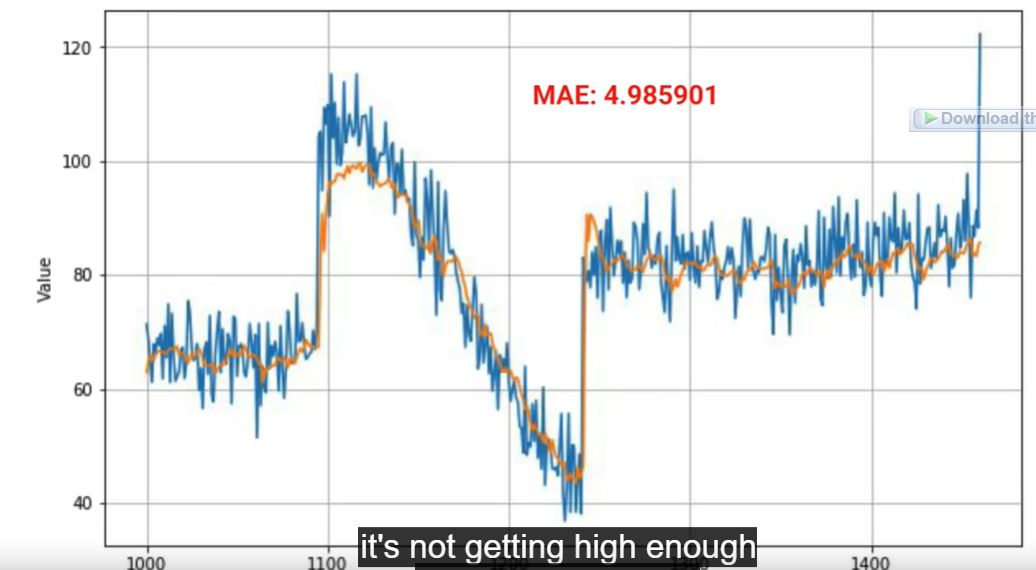
## Combining Convolutions with RNNs :

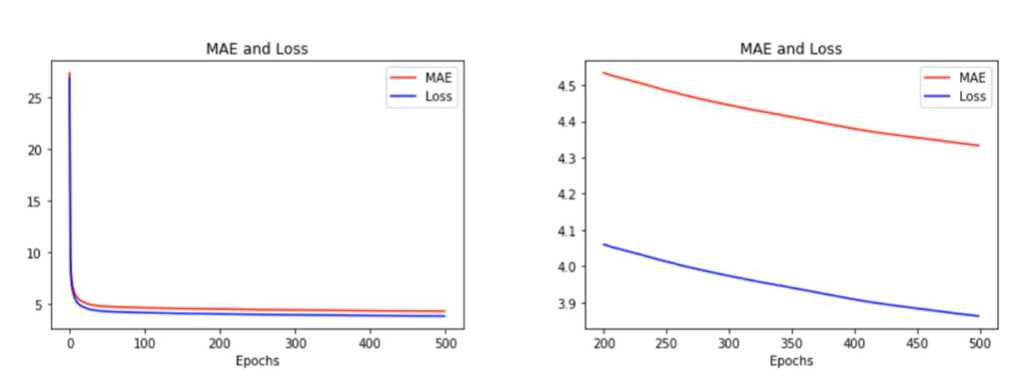
## 

#### Model evaluation :

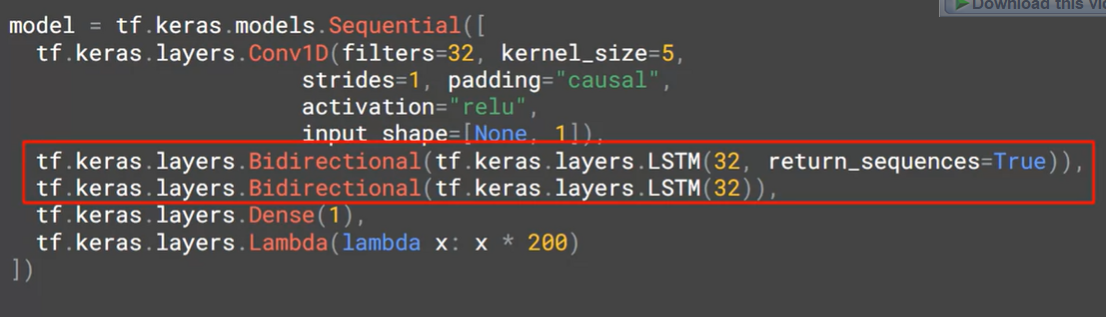


#### Possible solutions to make it better:

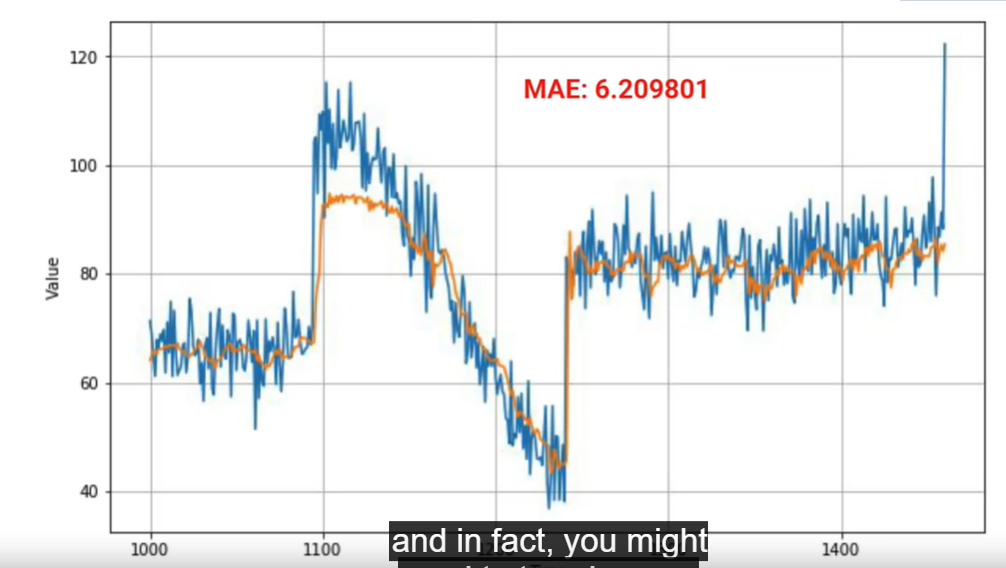
* Train for more epochs :



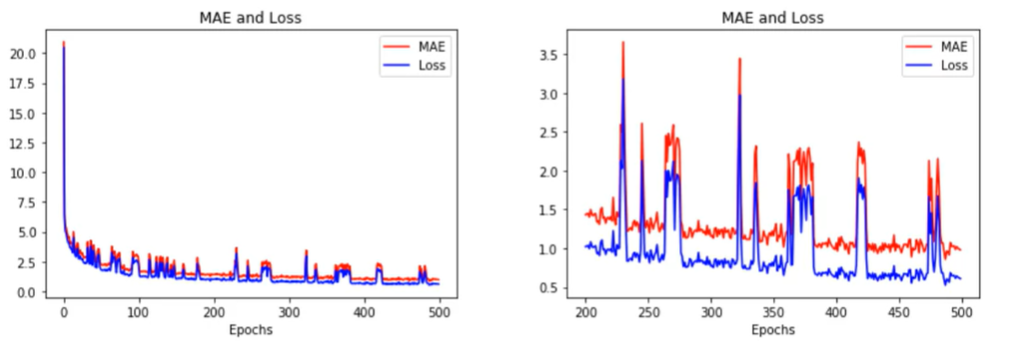
* We can see from the graph that the MAE and the loss are keep downgrading so training for more epochs would be interesting
* Make the LSTM bidirectional:



Result:



* While training , the MAZ was below 1 , but on the validation we are seeing a worse MAE : sign of overfitting
* There is also an obvious instability on the porgress of the MAE by epochs :

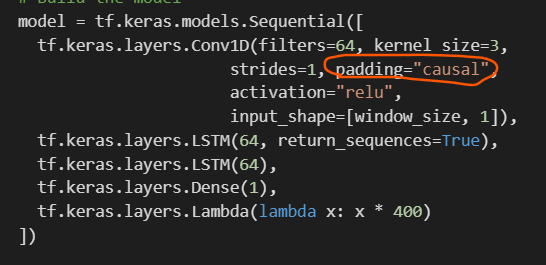


* The reason for these spikes can be the little size of the batching size ( which introduces further noise ) , soit’s worth to experiment with different batch size

More on batching size : <https://www.youtube.com/watch?v=4qJaSmvhxi8>

## Padding=”causal” in Conv1D is important for Time Series Prediction

* While doing prediction of **t** , the output shouldn’t be affected by the future values ( t+1 , t+2 , …etc )
  + Let’s say we have data = [ 1,2,3,4,5] and the Kernel size of Conv1D is 3
  + So in the first iteration ( t=1 ) : we will use [1,2(t=2),3(t=3)] to calculate it which isn’t a good practice while doing prediction
  + One possible solution is do left padding equal to 2 to the data : data=[0,0,1,2,3,4,5] :
    - The calculation of the first iteration now (t=1) will use [ 0,0,1] , so it will depend only to the actual value
    - For t=2 , it will use [0,1(t=1) , 2(t=2) ] , so it will depend only to the actual value and the previous one(s)
* This can be happened inside the Conv1D by adding the arg padding=” causal”

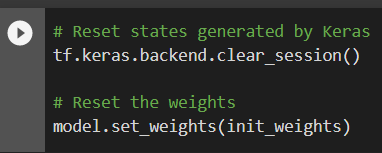


## Optional : What if we wanna get the same wights we had during the fine tuning process?

* In the previous labs, and while we were trying to get the perfect learning rate to train our model on it, we used two different models: one to get the perfect learning rate and another to do the real training
* If we wanna use the same model in both , we can acheuve it thanks to .get\_weights() method
* To get the weights after the first training ( to get the optimal learning rate ) :



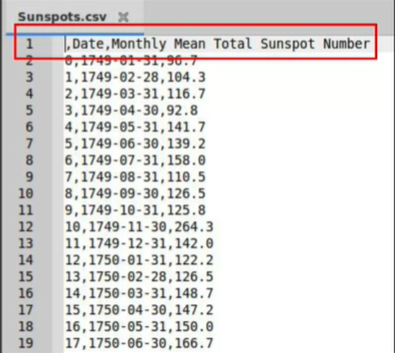
* To set the weights for the model to do our second training :

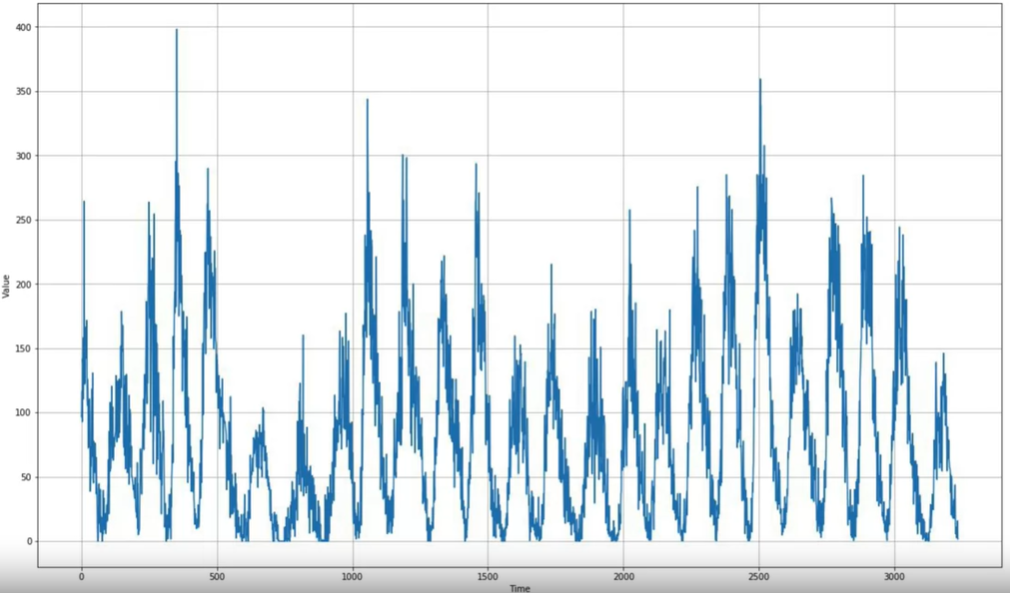


## Optional ( reminder ) : Adding callback for early stopping when the mae is good enough

In the bottom of this codelab <https://colab.research.google.com/github/https-deeplearning-ai/tensorflow-1-public/blob/main/C4/W4/ungraded_labs/C4_W4_Lab_1_LSTM.ipynb> we can say the way to stop the training once the Mae is below 5.2

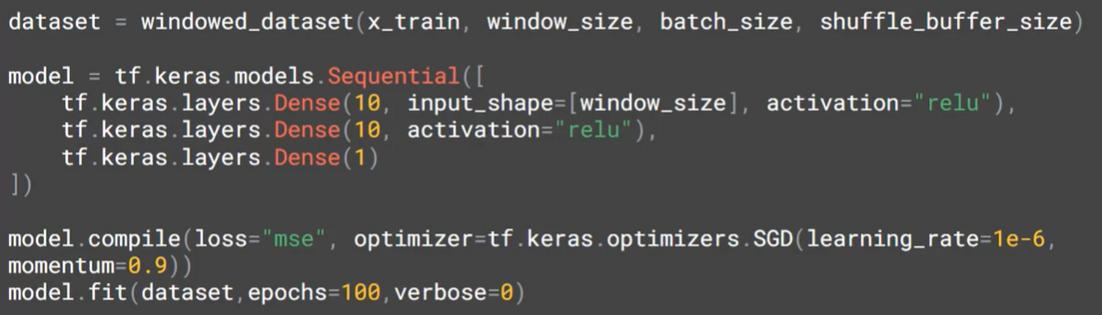
## Playing with real data ( Number of Sunspots ) :

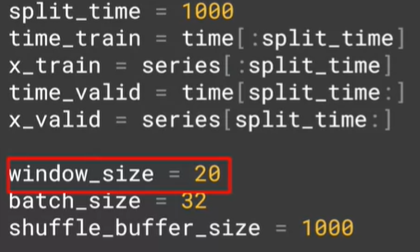




### Doing a DNN only architecture

#### Initial Architecture:





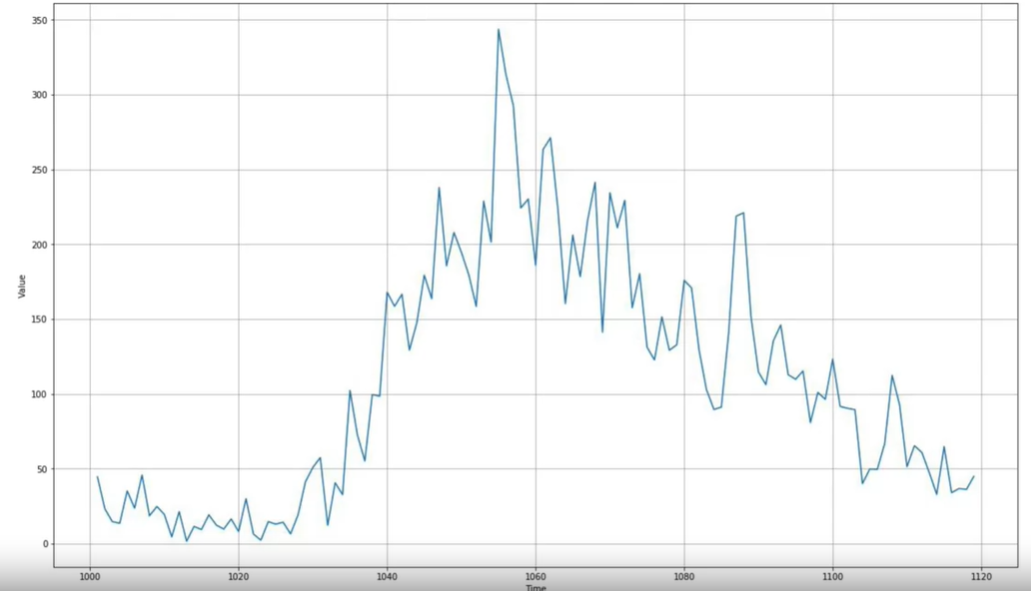
Result: MAE = 19.32

#### Potential solution: changing the window\_size

We are having window\_size=20 , and each timeslot presents a month , so the current window size is less than two years , but from the data chart , we notice that there is a seasonality each 11 years , so the window\_size should be equal to 132 ( = 11 years , the season )

Result: MAE = 23.54 (worse ! )

**Explanation:** even though the data is seasonal , but we don’t need to take the whole season in the window , because by zooming in , we can see that we have a typical timeseries where the data of the future depends on the data of the past ( with much noise ) :



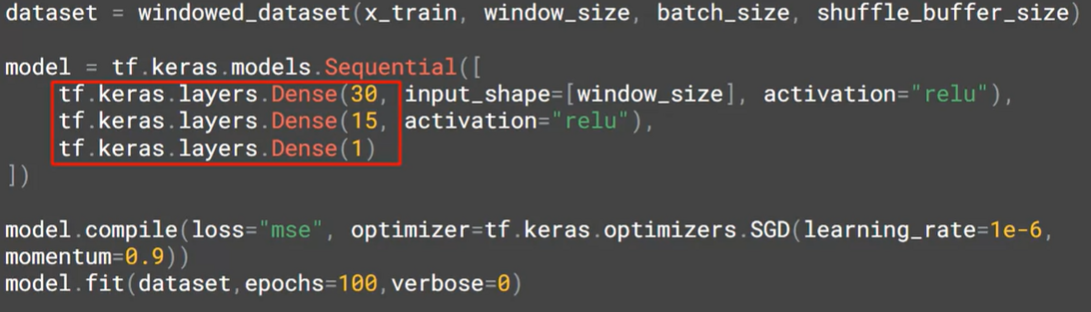
#### Potential solution: changing the split\_time :

the current configuration takes only 1000 timeslot to do the training and it lets 2500 timeslot to do the validation ! , where in reality , we should give more time to training

* Change the split\_time to 3000 ( only 500 timeslots for the validation ) with window\_size = 30

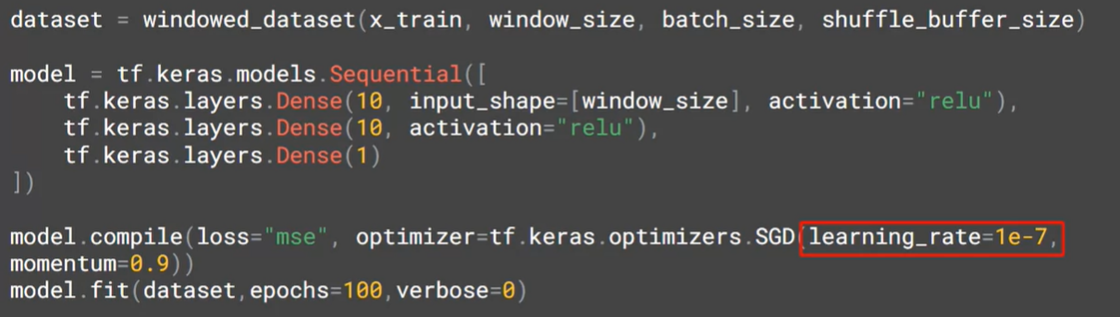
Result: MAE = 15.14 (better ! )

#### Potential Solution : Increasing the number of the units in the Dense layer



Result: MAE = 14.34 (little better ! )

#### Potential Solution : keeping the old number of units and Changing the learning rate



* We changed it from 10^-6 to 10^-7

Result: MAE = 14.12 (little better! )