An RNN model for temperature data

This time we will be working with real data: daily (Tmin, Tmax) temperature series from 36 weather stations spanning 50 years. It is to be noted that a pretty good predictor model already exists for temperatures: the average of temperatures on the same day of the year in N previous years. It is not clear if RNNs can do better but we will se how far they can go.

In [1]:
```
!pip install tensorflow==1.15.3
```

```
Collecting tensorflow==1.15.3
  Downloading tensorflow-1.15.3-cp37-cp37m-manylinux2010_x86_64.whl (110.5 MB)
     |████████████████████████████████| 110.5 MB 22 kB/s s eta 0:00:01
Collecting gast==0.2.2
  Downloading gast-0.2.2.tar.gz (10 kB)
Collecting tensorflow-estimator==1.15.1
  Downloading tensorflow_estimator-1.15.1-py2.py3-none-any.whl (503 kB)
     |████████████████████████████████| 503 kB 49.8 MB/s eta 0:00:01
Requirement already satisfied: protobuf>=3.6.1 in /opt/conda/lib/python3.7/site-
packages (from tensorflow==1.15.3) (3.18.1)
Requirement already satisfied: absl-py>=0.7.0 in /opt/conda/lib/python3.7/site-p
ackages (from tensorflow==1.15.3) (0.14.1)
Collecting keras-applications>=1.0.8
  Downloading Keras_Applications-1.0.8-py3-none-any.whl (50 kB)
     |████████████████████████████████| 50 kB 8.8 MB/s  eta 0:00:01
Requirement already satisfied: wrapt>=1.11.1 in /opt/conda/lib/python3.7/site-pa
ckages (from tensorflow==1.15.3) (1.13.1)
Requirement already satisfied: grpcio>=1.8.6 in /opt/conda/lib/python3.7/site-pa
ckages (from tensorflow==1.15.3) (1.38.1)
Requirement already satisfied: numpy<2.0,>=1.16.0 in /opt/conda/lib/python3.7/si
te-packages (from tensorflow==1.15.3) (1.19.5)
Requirement already satisfied: opt-einsum>=2.3.2 in /opt/conda/lib/python3.7/sit
e-packages (from tensorflow==1.15.3) (3.3.0)
Requirement already satisfied: six>=1.10.0 in /opt/conda/lib/python3.7/site-pack
ages (from tensorflow==1.15.3) (1.16.0)
Requirement already satisfied: google-pasta>=0.1.6 in /opt/conda/lib/python3.7/s
ite-packages (from tensorflow==1.15.3) (0.2.0)
Collecting tensorboard<1.16.0,>=1.15.0
  Downloading tensorboard-1.15.0-py3-none-any.whl (3.8 MB)
     |████████████████████████████████| 3.8 MB 16.4 MB/s eta 0:00:01
Requirement already satisfied: wheel>=0.26 in /opt/conda/lib/python3.7/site-pack
ages (from tensorflow==1.15.3) (0.37.0)
Collecting astor>=0.6.0
  Downloading astor-0.8.1-py2.py3-none-any.whl (27 kB)
Requirement already satisfied: termcolor>=1.1.0 in /opt/conda/lib/python3.7/site
-packages (from tensorflow==1.15.3) (1.1.0)
Requirement already satisfied: keras-preprocessing>=1.0.5 in /opt/conda/lib/pyth
on3.7/site-packages (from tensorflow==1.15.3) (1.1.2)
Requirement already satisfied: h5py in /opt/conda/lib/python3.7/site-packages (f
rom keras-applications>=1.0.8->tensorflow==1.15.3) (3.1.0)
Requirement already satisfied: markdown>=2.6.8 in /opt/conda/lib/python3.7/site-
packages (from tensorboard<1.16.0,>=1.15.0->tensorflow==1.15.3) (3.3.4)
Requirement already satisfied: werkzeug>=0.11.15 in /opt/conda/lib/python3.7/sit
e-packages (from tensorboard<1.16.0,>=1.15.0->tensorflow==1.15.3) (2.0.2)
Requirement already satisfied: setuptools>=41.0.0 in /opt/conda/lib/python3.7/si
te-packages (from tensorboard<1.16.0,>=1.15.0->tensorflow==1.15.3) (58.2.0)
Requirement already satisfied: importlib-metadata in /opt/conda/lib/python3.7/si
```

```
te-packages (from markdown>=2.6.8->tensorboard<1.16.0,>=1.15.0->tensorflow==1.1
5.3) (4.8.1)
Requirement already satisfied: cached-property in /opt/conda/lib/python3.7/site-
packages (from h5py->keras-applications>=1.0.8->tensorflow==1.15.3) (1.5.2)
Requirement already satisfied: typing-extensions>=3.6.4 in /opt/conda/lib/python
3.7/site-packages (from importlib-metadata->markdown>=2.6.8->tensorboard<1.16.0,
>=1.15.0->tensorflow==1.15.3) (3.10.0.2)
Requirement already satisfied: zipp>=0.5 in /opt/conda/lib/python3.7/site-packag
es (from importlib-metadata->markdown>=2.6.8->tensorboard<1.16.0,>=1.15.0->tenso
rflow==1.15.3) (3.6.0)
Building wheels for collected packages: gast
  Building wheel for gast (setup.py) ... done
  Created wheel for gast: filename=gast-0.2.2-py3-none-any.whl size=7554 sha256=
08cb902cd0c6bdff6f5fc7731880608f137a80b3322f81bdc49678468b4b8b8b
  Stored in directory: /home/jupyter/.cache/pip/wheels/21/7f/02/420f32a803f7d096
7b48dd823da3f558c5166991bfd204eef3
Successfully built gast
Installing collected packages: tensorflow-estimator, tensorboard, keras-applicat
ions, gast, astor, tensorflow
  Attempting uninstall: tensorflow-estimator
    Found existing installation: tensorflow-estimator 2.6.0
    Uninstalling tensorflow-estimator-2.6.0:
      Successfully uninstalled tensorflow-estimator-2.6.0
  Attempting uninstall: tensorboard
    Found existing installation: tensorboard 2.5.0
    Uninstalling tensorboard-2.5.0:
      Successfully uninstalled tensorboard-2.5.0
  Attempting uninstall: gast
    Found existing installation: gast 0.4.0
    Uninstalling gast-0.4.0:
      Successfully uninstalled gast-0.4.0
  Attempting uninstall: tensorflow
    Found existing installation: tensorflow 2.6.0
    Uninstalling tensorflow-2.6.0:
      Successfully uninstalled tensorflow-2.6.0
ERROR: pip's dependency resolver does not currently take into account all the pa
ckages that are installed. This behaviour is the source of the following depende
ncy conflicts.
tensorflow-io 0.18.0 requires tensorflow-io-gcs-filesystem==0.18.0, which is not
installed.
explainable-ai-sdk 1.3.2 requires xai-image-widget, which is not installed.
tfx-bsl 1.3.0 requires absl-py<0.13,>=0.9, but you have absl-py 0.14.1 which is
 incompatible.
tfx-bsl 1.3.0 requires google-api-python-client<2,>=1.7.11, but you have google-
api-python-client 2.24.0 which is incompatible.
tfx-bsl 1.3.0 requires pyarrow<3,>=1, but you have pyarrow 5.0.0 which is incomp
atible.
tensorflow-transform 1.3.0 requires absl-py<0.13,>=0.9, but you have absl-py 0.1
4.1 which is incompatible.
tensorflow-transform 1.3.0 requires pyarrow<3,>=1, but you have pyarrow 5.0.0 wh
ich is incompatible.
tensorflow-serving-api 2.6.0 requires tensorflow<3,>=2.6.0, but you have tensorf
low 1.15.3 which is incompatible.
tensorflow-probability 0.13.0rc0 requires gast>=0.3.2, but you have gast 0.2.2 w
hich is incompatible.
tensorflow-io 0.18.0 requires tensorflow<2.6.0,>=2.5.0, but you have tensorflow
 1.15.3 which is incompatible.
tensorflow-cloud 0.1.14 requires tensorboard>=2.3.0, but you have tensorboard 1.
15.0 which is incompatible.
Successfully installed astor-0.8.1 gast-0.2.2 keras-applications-1.0.8 tensorboa
rd-1.15.0 tensorflow-1.15.3 tensorflow-estimator-1.15.1
```

Please ignore any compatibility warnings and errors. Make sure to **restart** your kernel to ensure this change has taken place.

In [1]:
```
import math
import sys
import time
import numpy as np

sys.path.insert(0, 'temperatures/utils/') #so python can find the utils_ modules
import utils_batching
import utils_args

import tensorflow as tf
from tensorflow.python.lib.io import file_io as gfile
print("Tensorflow version: " + tf.__version__)
```

```
Tensorflow version: 1.15.3
```

In [2]:
```
from matplotlib import pyplot as plt
import utils_prettystyle
import utils_display
```

Download Data

In [3]:
```
%%bash
DOWNLOAD_DIR=temperatures/data
mkdir $DOWNLOAD_DIR
gsutil -m cp gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequ
```

```
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USC00025412.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USC00054452.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USC00033862.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USC00049742.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USC00026132.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USC00055722.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00013870.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00014860.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00014850.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00014820.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USC00091732.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00013880.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USC00087982.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
```

```
temperatures/USW00003820.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00013960.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00013970.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00014840.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00013740.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00014740.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00014750.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00014940.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00014950.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00023050.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00023070.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00023160.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00023170.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00024090.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00024130.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00024150.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00026510.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00026410.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00093720.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00093820.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00094830.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00094910.csv...
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
temperatures/USW00093730.csv...
- [36/36 files][ 14.2 MiB/ 14.2 MiB] 100% Done
Operation completed over 36 objects/14.2 MiB.
```

Hyperparameters

N_FORWARD = 1: works but model struggles to predict from some positions

N_FORWARD = 4: better but still bad occasionnally

N_FORWARD = 8: works perfectly

In [4]:

```
NB_EPOCHS = 5       # number of times the model sees all the data during trainin

N_FORWARD = 8       # train the network to predict N in advance (traditionnally
RESAMPLE_BY = 5     # averaging period in days (training on daily data is too mu
```

```
RNN_CELLSIZE = 128  # size of the RNN cells
N_LAYERS = 2        # number of stacked RNN cells (needed for tensor shapes but
SEQLEN = 128        # unrolled sequence length
BATCHSIZE = 64      # mini-batch size
DROPOUT_PKEEP = 0.7 # probability of neurons not being dropped (should be betwee
ACTIVATION = tf.nn.tanh # Activation function for GRU cells (tf.nn.relu or tf.nn

JOB_DIR  = "checkpoints"
DATA_DIR = "temperatures/data"

# potentially override some settings from command-line arguments
if __name__ == '__main__':
    JOB_DIR, DATA_DIR = utils_args.read_args1(JOB_DIR, DATA_DIR)

ALL_FILEPATTERN = DATA_DIR + "/*.csv" # pattern matches all 1666 files
EVAL_FILEPATTERN = DATA_DIR + "/USC000*2.csv" # pattern matches 8 files
# pattern USW*.csv -> 298 files, pattern USW*0.csv -> 28 files
print('Reading data from "{}".\nWrinting checkpoints to "{}".'.format(DATA_DIR,
```

```
Reading data from "temperatures/data".
Wrinting checkpoints to "checkpoints".
```

Temperature data

This is what our temperature datasets looks like: sequences of daily (Tmin, Tmax) from 1960 to 2010. They have been cleaned up and eventual missing values have been filled by interpolation. Interpolated regions of the dataset are marked in red on the graph.

In [5]:

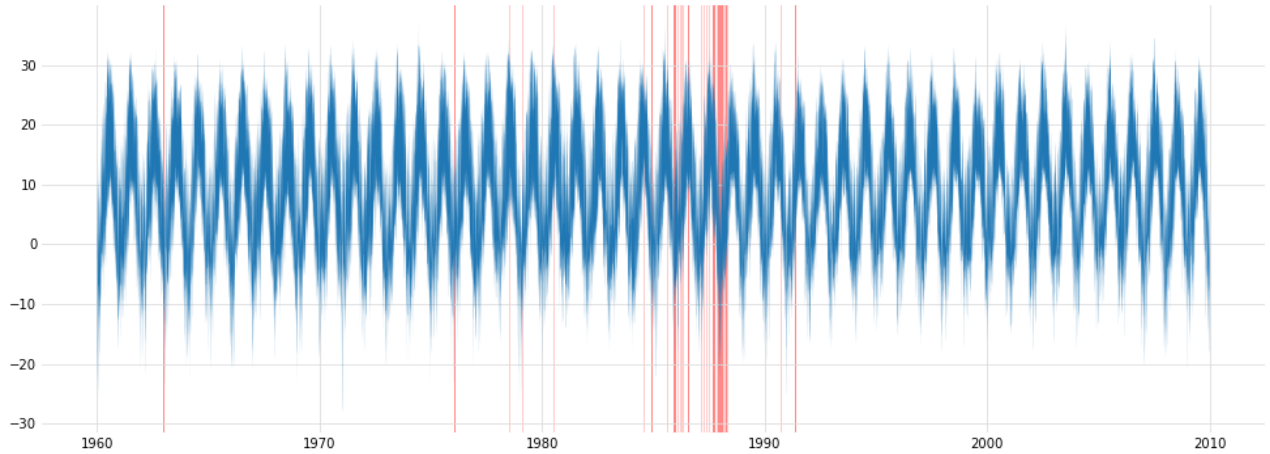
```
all_filenames = gfile.get_matching_files(ALL_FILEPATTERN)
eval_filenames = gfile.get_matching_files(EVAL_FILEPATTERN)
train_filenames = list(set(all_filenames) - set(eval_filenames))

# By default, this utility function loads all the files and places data
# from them as-is in an array, one file per line. Later, we will use it
# to shape the dataset as needed for training.
ite = utils_batching.rnn_multistation_sampling_temperature_sequencer(eval_filena
evtemps, _, evdates, _, _ = next(ite) # gets everything

print('Pattern "{}" matches {} files'.format(ALL_FILEPATTERN, len(all_filenames)
print('Pattern "{}" matches {} files'.format(EVAL_FILEPATTERN, len(eval_filename
print("Evaluation files: {}".format(len(eval_filenames)))
print("Training files: {}".format(len(train_filenames)))
print("Initial shape of the evaluation dataset: " + str(evtemps.shape))
print("{} files, {} data points per file, {} values per data point"
      " (Tmin, Tmax, is_interpolated) ".format(evtemps.shape[0], evtemps.shape[1
```

```
Loading 8 files........
Pattern "temperatures/data/*.csv" matches 36 files
Pattern "temperatures/data/USC000*2.csv" matches 8 files
Evaluation files: 8
Training files: 28
Initial shape of the evaluation dataset: (8, 18262, 3)
8 files, 18262 data points per file, 3 values per data point (Tmin, Tmax, is_int
erpolated)
```

In [6]:

```
# You can adjust the visualisation range and dataset here.
# Interpolated regions of the dataset are marked in red.
```

```
WEATHER_STATION = 0 # 0 to 7 in default eval dataset
START_DATE = 0      # 0 = Jan 2nd 1950
END_DATE = 18262    # 18262 = Dec 31st 2009
visu_temperatures = evtemps[WEATHER_STATION,START_DATE:END_DATE]
visu_dates = evdates[START_DATE:END_DATE]

utils_display.picture_this_4(visu_temperatures, visu_dates)
```



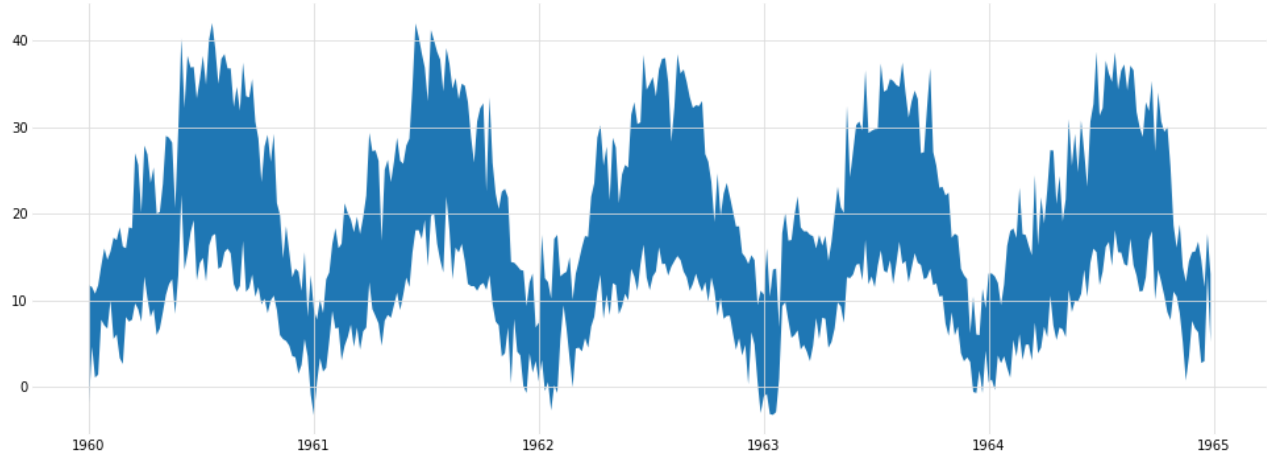
Resampling

Our RNN would need ot be unrolled across 365 steps to capture the yearly temperature cycles. That's a bit too much. We will resample the temparatures and work with 5-day averages for example. This is what resampled (Tmin, Tmax) temperatures look like.

In [7]:
```
# This time we ask the utility function to average temperatures over 5-day perio
ite = utils_batching.rnn_multistation_sampling_temperature_sequencer(eval_filena
evaltemps, _, evaldates, _, _ = next(ite)
```

```
Loading 8 files........
```

In [8]:
```
# display five years worth of data
WEATHER_STATION = 0              # 0 to 7 in default eval dataset
START_DATE = 0                   # 0 = Jan 2nd 1950
END_DATE = 365*5//RESAMPLE_BY    # 5 years
visu_temperatures = evaltemps[WEATHER_STATION, START_DATE:END_DATE]
visu_dates = evaldates[START_DATE:END_DATE]
plt.fill_between(visu_dates, visu_temperatures[:,0], visu_temperatures[:,1])
plt.show()
```



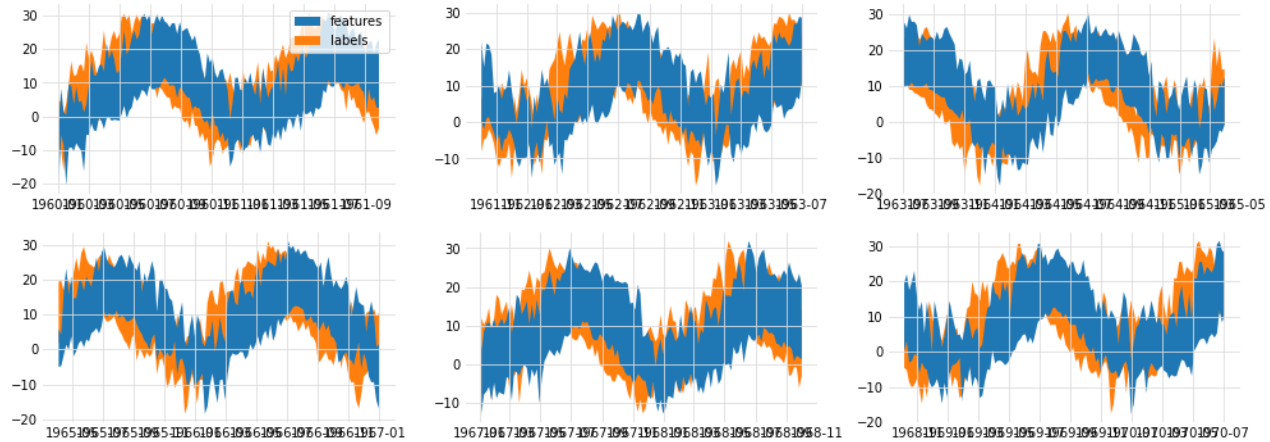
Visualize training sequences

This is what the neural network will see during training.

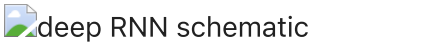
In [9]:
```
# The function rnn_multistation_sampling_temperature_sequencer puts one weather
# a batch and continues with data from the same station in corresponding lines i
# Features and labels are returned with shapes [BATCHSIZE, SEQLEN, 2]. The last
# contains (Tmin, Tmax).
ite = utils_batching.rnn_multistation_sampling_temperature_sequencer(eval_filena
                                                                     RESAMPLE_BY
                                                                     BATCHSIZE,
                                                                     SEQLEN,
                                                                     N_FORWARD,
                                                                     nb_epochs=1
                                                                     tminmax=Tru

# load 6 training sequences (each one contains data for all weather stations)
visu_data = [next(ite) for _ in range(6)]
```

```
Loading 8 files........
```

In [10]:
```
# Check that consecutive training sequences from the same weather station are in
WEATHER_STATION = 4
utils_display.picture_this_5(visu_data, WEATHER_STATION)
```



The model definition




```
                                                X shape [BATCHSIZE, SEQLEN, 2]
                                                Y shape [BATCHSIZE, SEQLEN, 2]
                                    H shape [BATCHSIZE, RNN_CELLSIZE*NLAYERS]
```

When executed, this function instantiates the Tensorflow graph for our model.

In [11]:
```
def model_rnn_fn(features, Hin, labels, step, dropout_pkeep):
    X = features  # shape [BATCHSIZE, SEQLEN, 2], 2 for (Tmin, Tmax)
    batchsize = tf.shape(X)[0]
    seqlen = tf.shape(X)[1]
    pairlen = tf.shape(X)[2] # should be 2 (tmin, tmax)

    cells = [tf.nn.rnn_cell.GRUCell(RNN_CELLSIZE, activation=ACTIVATION) for _ i
    # dropout useful between cell layers only: no output dropout on last cell
    cells = [tf.nn.rnn_cell.DropoutWrapper(cell, output_keep_prob = dropout_pkee
    # a stacked RNN cell still works like an RNN cell
    cell = tf.nn.rnn_cell.MultiRNNCell(cells, state_is_tuple=False)
    # X[BATCHSIZE, SEQLEN, 2], Hin[BATCHSIZE, RNN_CELLSIZE*N_LAYERS]
    # the sequence unrolling happens here
    Yn, H = tf.nn.dynamic_rnn(cell, X, initial_state=Hin, dtype=tf.float32)
    # Yn[BATCHSIZE, SEQLEN, RNN_CELLSIZE]
    Yn = tf.reshape(Yn, [batchsize*seqlen, RNN_CELLSIZE])
    Yr = tf.layers.dense(Yn, 2) # Yr [BATCHSIZE*SEQLEN, 2]
    Yr = tf.reshape(Yr, [batchsize, seqlen, 2]) # Yr [BATCHSIZE, SEQLEN, 2]
    Yout = Yr[:,-N_FORWARD:,:] # Last N_FORWARD outputs Yout [BATCHSIZE, N_FORWA

    loss = tf.losses.mean_squared_error(Yr, labels) # labels[BATCHSIZE, SEQLEN,

    lr = 0.001 + tf.train.exponential_decay(0.01, step, 1000, 0.5)
    optimizer = tf.train.AdamOptimizer(learning_rate=lr)
    train_op = optimizer.minimize(loss)

    return Yout, H, loss, train_op, Yr
```

Instantiate the model

In [12]:
```
tf.reset_default_graph() # restart model graph from scratch

# placeholder for inputs
Hin = tf.placeholder(tf.float32, [None, RNN_CELLSIZE * N_LAYERS])
features = tf.placeholder(tf.float32, [None, None, 2]) # [BATCHSIZE, SEQLEN, 2]
labels = tf.placeholder(tf.float32, [None, None, 2]) # [BATCHSIZE, SEQLEN, 2]
step = tf.placeholder(tf.int32)
dropout_pkeep = tf.placeholder(tf.float32)

# instantiate the model
Yout, H, loss, train_op, Yr = model_rnn_fn(features, Hin, labels, step, dropout_
```

```
WARNING:tensorflow:From /tmp/ipykernel_25662/71392446.py:1: The name tf.reset_de
fault_graph is deprecated. Please use tf.compat.v1.reset_default_graph instead.

WARNING:tensorflow:From /tmp/ipykernel_25662/71392446.py:4: The name tf.placehol
der is deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /tmp/ipykernel_25662/2393688187.py:7: GRUCell.__init__
```

```
(from tensorflow.python.ops.rnn_cell_impl) is deprecated and will be removed in
a future version.
Instructions for updating:
This class is equivalent as tf.keras.layers.GRUCell, and will be replaced by tha
t in Tensorflow 2.0.
WARNING:tensorflow:From /tmp/ipykernel_25662/2393688187.py:9: The name tf.nn.rnn
_cell.DropoutWrapper is deprecated. Please use tf.compat.v1.nn.rnn_cell.DropoutW
rapper instead.

WARNING:tensorflow:From /tmp/ipykernel_25662/2393688187.py:11: MultiRNNCell.__in
it__ (from tensorflow.python.ops.rnn_cell_impl) is deprecated and will be remove
d in a future version.
Instructions for updating:
This class is equivalent as tf.keras.layers.StackedRNNCells, and will be replace
d by that in Tensorflow 2.0.
WARNING:tensorflow:From /tmp/ipykernel_25662/2393688187.py:14: dynamic_rnn (from
tensorflow.python.ops.rnn) is deprecated and will be removed in a future versio
n.
Instructions for updating:
Please use `keras.layers.RNN(cell)`, which is equivalent to this API
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/p
ython/ops/rnn_cell_impl.py:559: Layer.add_variable (from tensorflow.python.kera
s.engine.base_layer) is deprecated and will be removed in a future version.
Instructions for updating:
Please use `layer.add_weight` method instead.
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/p
ython/ops/rnn_cell_impl.py:565: calling Constant.__init__ (from tensorflow.pytho
n.ops.init_ops) with dtype is deprecated and will be removed in a future versio
n.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the c
onstructor
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/p
ython/ops/rnn_cell_impl.py:575: calling Zeros.__init__ (from tensorflow.python.o
ps.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the c
onstructor
WARNING:tensorflow:From /tmp/ipykernel_25662/2393688187.py:17: dense (from tenso
rflow.python.layers.core) is deprecated and will be removed in a future version.
Instructions for updating:
Use keras.layers.Dense instead.
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/p
ython/layers/core.py:187: Layer.apply (from tensorflow.python.keras.engine.base_
layer) is deprecated and will be removed in a future version.
Instructions for updating:
Please use `layer.__call__` method instead.
WARNING:tensorflow:From /tmp/ipykernel_25662/2393688187.py:21: The name tf.losse
s.mean_squared_error is deprecated. Please use tf.compat.v1.losses.mean_squared_
error instead.

WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/p
ython/ops/losses/losses_impl.py:121: where (from tensorflow.python.ops.array_op
s) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /tmp/ipykernel_25662/2393688187.py:23: The name tf.trai
n.exponential_decay is deprecated. Please use tf.compat.v1.train.exponential_dec
ay instead.

WARNING:tensorflow:From /tmp/ipykernel_25662/2393688187.py:24: The name tf.trai
```

```
n.AdamOptimizer is deprecated. Please use tf.compat.v1.train.AdamOptimizer inste
ad.
```

Initialize Tensorflow session

This resets all neuron weights and biases to initial random values

In [13]:
```
# variable initialization
sess = tf.Session()
init = tf.global_variables_initializer()
sess.run([init])
saver = tf.train.Saver(max_to_keep=1)
```

```
WARNING:tensorflow:From /tmp/ipykernel_25662/1210481336.py:2: The name tf.Sessio
n is deprecated. Please use tf.compat.v1.Session instead.

WARNING:tensorflow:From /tmp/ipykernel_25662/1210481336.py:3: The name tf.global
_variables_initializer is deprecated. Please use tf.compat.v1.global_variables_i
nitializer instead.

WARNING:tensorflow:From /tmp/ipykernel_25662/1210481336.py:5: The name tf.train.
Saver is deprecated. Please use tf.compat.v1.train.Saver instead.

2021-10-25 19:22:59.059977: I tensorflow/core/platform/cpu_feature_guard.cc:142]
Your CPU supports instructions that this TensorFlow binary was not compiled to u
se: AVX2 FMA
2021-10-25 19:22:59.068433: I tensorflow/core/platform/profile_utils/cpu_utils.c
c:94] CPU Frequency: 2199995000 Hz
2021-10-25 19:22:59.069358: I tensorflow/compiler/xla/service/service.cc:168] XL
A service 0x562186374cd0 initialized for platform Host (this does not guarantee
that XLA will be used). Devices:
2021-10-25 19:22:59.069405: I tensorflow/compiler/xla/service/service.cc:176]
StreamExecutor device (0): Host, Default Version
```

The training loop

You can re-execute this cell to continue training.

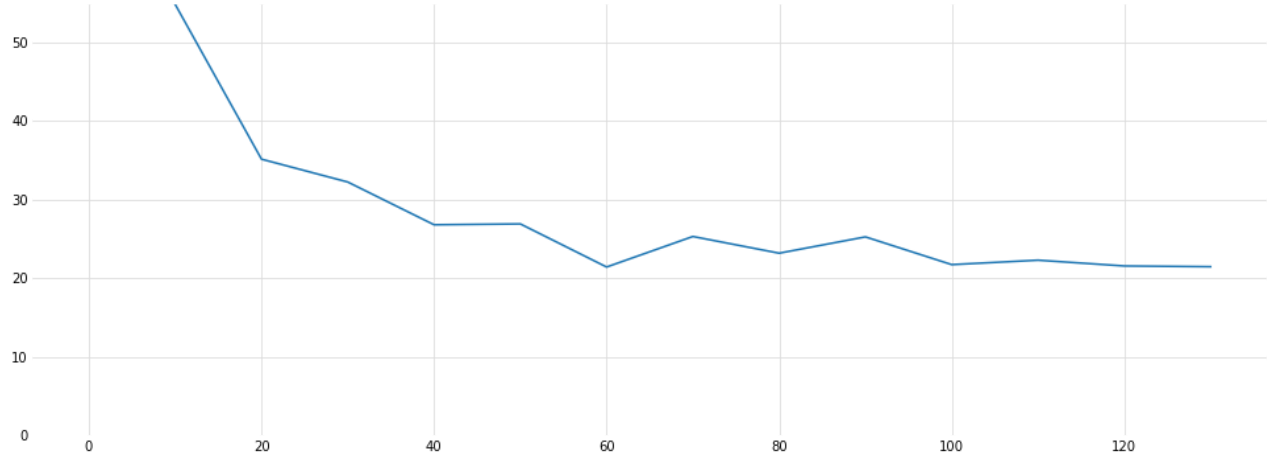
Training data must be batched correctly, one weather station per line, continued on the same line across batches. This way, output states computed from one batch are the correct input states for the next batch. The provided utility function `rnn_multistation_sampling_temperature_sequencer` does the right thing.
batching for RNNs

In [14]:
```
losses = []
indices = []
last_epoch = 99999
last_fileid = 99999

for i, (next_features, next_labels, dates, epoch, fileid) in enumerate(
    utils_batching.rnn_multistation_sampling_temperature_sequencer(train_filenam
                                                                   RESAMPLE_BY,
                                                                   BATCHSIZE,
```

```
                                                                    SEQLEN,
                                                                    N_FORWARD,
                                                                    NB_EPOCHS, tm

    # reinintialize state between epochs or when starting on data from a new wea
    if epoch != last_epoch or fileid != last_fileid:
        batchsize = next_features.shape[0]
        H_ = np.zeros([batchsize, RNN_CELLSIZE * N_LAYERS])
        print("State reset")

    #train
    feed = {Hin: H_, features: next_features, labels: next_labels, step: i, drop
    Yout_, H_, loss_, _, Yr_ = sess.run([Yout, H, loss, train_op, Yr], feed_dict

    # print progress
    if i%20 == 0:
        print("{}: epoch {} loss = {} ({} weather stations this epoch)".format(i
        sys.stdout.flush()
    if i%10 == 0:
        losses.append(np.mean(loss_))
        indices.append(i)

    last_epoch = epoch
    last_fileid = fileid

# save the trained model
SAVEDMODEL = JOB_DIR + "/ckpt" + str(int(time.time()))
tf.saved_model.simple_save(sess, SAVEDMODEL,
                           inputs={"features":features, "Hin":Hin, "dropout_pkee
                           outputs={"Yout":Yout, "H":H})
```

```
Loading 28 files.............................
State reset
0: epoch 0 loss = 294.3005676269531 (28 weather stations this epoch)
20: epoch 0 loss = 35.1406364440918 (28 weather stations this epoch)
State reset
40: epoch 1 loss = 26.80422019958496 (28 weather stations this epoch)
State reset
60: epoch 2 loss = 21.43130111694336 (28 weather stations this epoch)
80: epoch 2 loss = 23.1910400390625 (28 weather stations this epoch)
State reset
100: epoch 3 loss = 21.73704719543457 (28 weather stations this epoch)
State reset
120: epoch 4 loss = 21.566041946411133 (28 weather stations this epoch)
WARNING:tensorflow:From /tmp/ipykernel_25662/347898.py:39: simple_save (from ten
sorflow.python.saved_model.simple_save) is deprecated and will be removed in a f
uture version.
Instructions for updating:
This function will only be available through the v1 compatibility library as tf.
compat.v1.saved_model.simple_save.
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/p
ython/saved_model/signature_def_utils_impl.py:201: build_tensor_info (from tenso
rflow.python.saved_model.utils_impl) is deprecated and will be removed in a futu
re version.
Instructions for updating:
This function will only be available through the v1 compatibility library as tf.
compat.v1.saved_model.utils.build_tensor_info or tf.compat.v1.saved_model.build_
tensor_info.
INFO:tensorflow:Assets added to graph.
INFO:tensorflow:No assets to write.
INFO:tensorflow:SavedModel written to: checkpoints/ckpt1635189825/saved_model.pb
```

In [15]:
```
plt.ylim(ymax=np.amax(losses[1:])) # ignore first value for scaling
plt.plot(indices, losses)
plt.show()
```



Inference

This is a generative model: run an trained RNN cell in a loop

In [16]:
```
def prediction_run(predict_fn, prime_data, run_length):
    H = np.zeros([1, RNN_CELLSIZE * N_LAYERS]) # zero state initially
    Yout = np.zeros([1, N_FORWARD, 2])
    data_len = prime_data.shape[0]-N_FORWARD

    # prime the state from data
    if data_len > 0:
        Yin = np.array(prime_data[:-N_FORWARD])
        Yin = np.reshape(Yin, [1, data_len, 2]) # reshape as one sequence of pai
        r = predict_fn({'features': Yin, 'Hin':H, 'dropout_pkeep':1.0}) # no dro
        Yout = r["Yout"]
        H = r["H"]

        # initaily, put real data on the inputs, not predictions
        Yout = np.expand_dims(prime_data[-N_FORWARD:], axis=0)
        # Yout shape [1, N_FORWARD, 2]: batch of a single sequence of length N_F

    # run prediction
    # To generate a sequence, run a trained cell in a loop passing as input and
    # respectively the output and output state from the previous iteration.
    results = []
    for i in range(run_length//N_FORWARD+1):
        r = predict_fn({'features': Yout, 'Hin':H, 'dropout_pkeep':1.0}) # no dr
        Yout = r["Yout"]
        H = r["H"]
        results.append(Yout[0]) # shape [N_FORWARD, 2]

    return np.concatenate(results, axis=0)[:run_length]
```

Validation

In [17]:

```
QYEAR = 365//(RESAMPLE_BY*4)
YEAR = 365//(RESAMPLE_BY)

# Try starting predictions from January / March / July (resp. OFFSET = YEAR or Y
# Some start dates are more challenging for the model than others.
OFFSET = 30*YEAR+1*QYEAR

PRIMELEN=5*YEAR
RUNLEN=3*YEAR
RMSELEN=3*365//(RESAMPLE_BY*2) # accuracy of predictions 1.5 years in advance
```

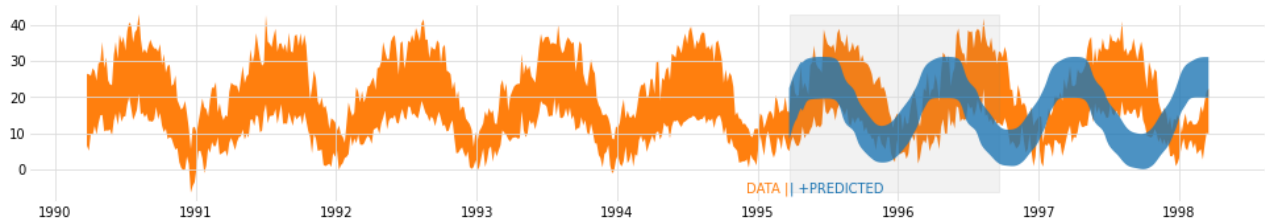
In [18]:

```
# Restore the model from the last checkpoint saved previously.

# Alternative checkpoints:
# Once you have trained on all 1666 weather stations on Google Cloud ML Engine,
# SAVEDMODEL = "gs://{BUCKET}/sinejobs/sines_XXXXXX_XXXXXX/ckptXXXXXXXX"
# A sample checkpoint is provided with the lab. You can try loading it for compa
# SAVEDMODEL = "temperatures_best_checkpoint"

predict_fn = tf.contrib.predictor.from_saved_model(SAVEDMODEL)
```

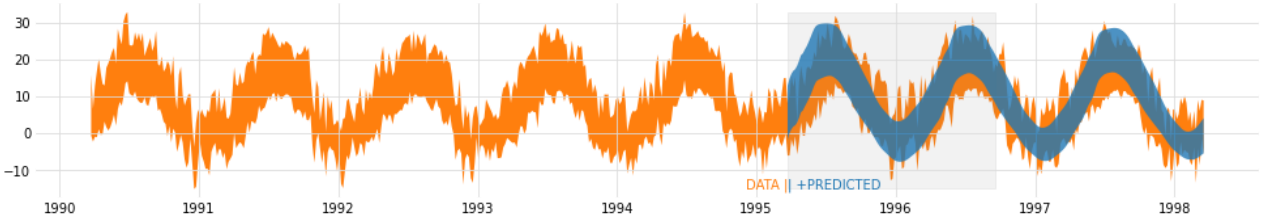
```
WARNING:tensorflow:
The TensorFlow contrib module will not be included in TensorFlow 2.0.
For more information, please see:
  * https://github.com/tensorflow/community/blob/master/rfcs/20180907-contrib-su
nset.md
  * https://github.com/tensorflow/addons
  * https://github.com/tensorflow/io (for I/O related ops)
If you depend on functionality not listed there, please file an issue.

WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow_core/c
ontrib/predictor/saved_model_predictor.py:153: load (from tensorflow.python.save
d_model.loader_impl) is deprecated and will be removed in a future version.
Instructions for updating:
This function will only be available through the v1 compatibility library as tf.
compat.v1.saved_model.loader.load or tf.compat.v1.saved_model.load. There will b
e a new function for importing SavedModels in Tensorflow 2.0.
INFO:tensorflow:Restoring parameters from checkpoints/ckpt1635189825/variables/v
ariables
```

In [19]:

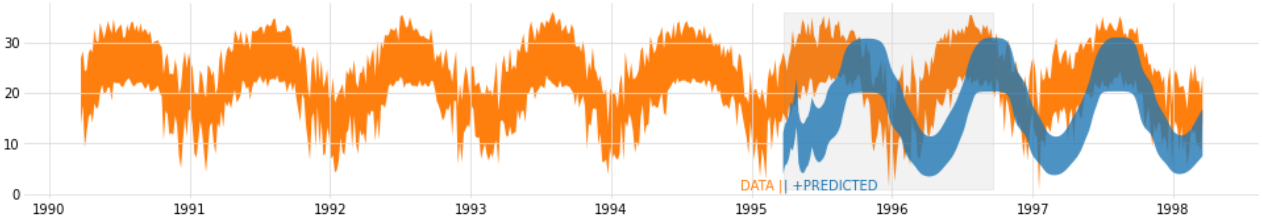
```
for evaldata in evaltemps:
    prime_data = evaldata[OFFSET:OFFSET+PRIMELEN]
    results = prediction_run(predict_fn, prime_data, RUNLEN)
    utils_display.picture_this_6(evaldata, evaldates, prime_data, results, PRIME
```



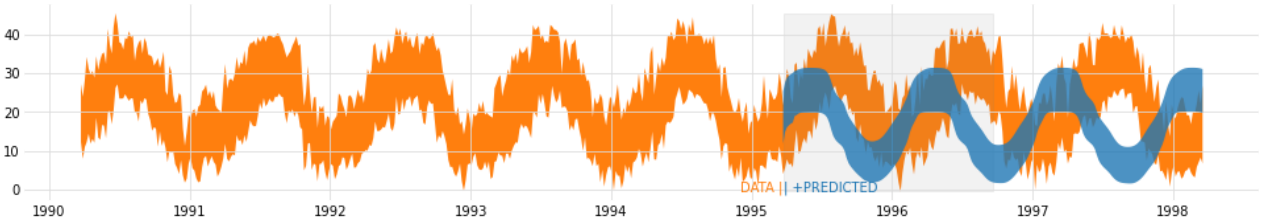

```
RMSE on 109 predictions (shaded area): 8.488105536032977
```



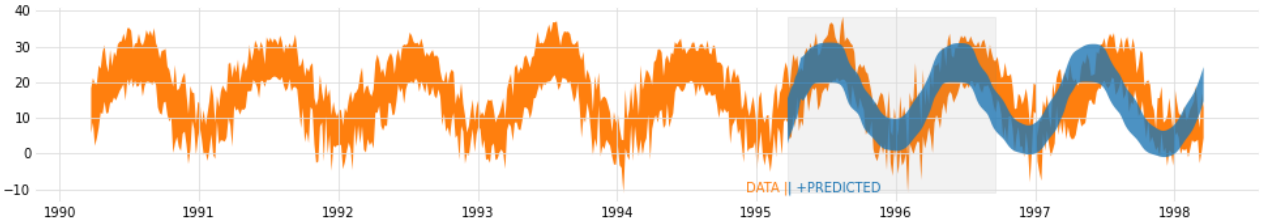

```
RMSE on 109 predictions (shaded area): 4.599209103638949
```



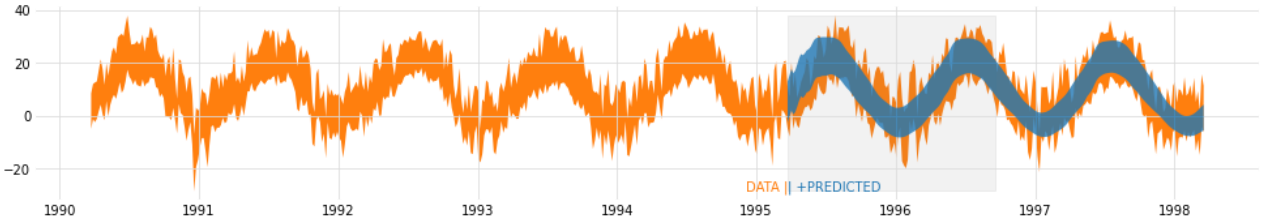

```
RMSE on 109 predictions (shaded area): 10.158704121455557
```



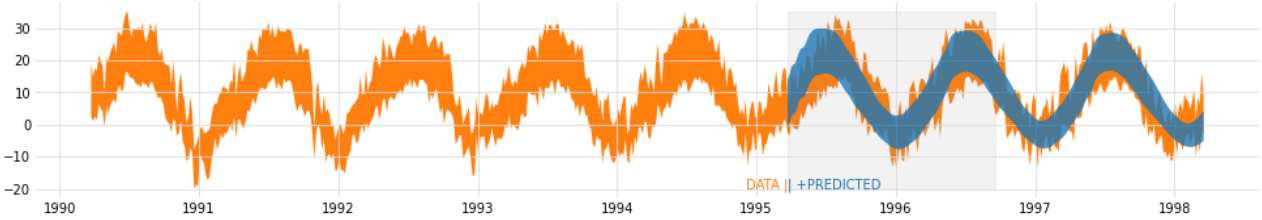

```
RMSE on 109 predictions (shaded area): 11.773308889245957
```



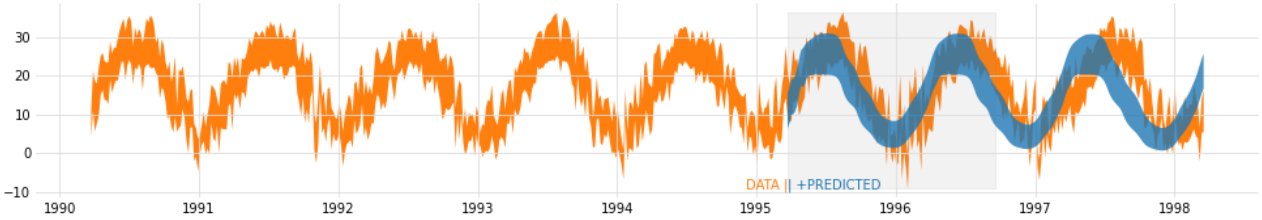

```
RMSE on 109 predictions (shaded area): 4.778364026688139
```




```
RMSE on 109 predictions (shaded area): 5.6760404453718785
```




```
RMSE on 109 predictions (shaded area): 4.491668121229426
```




```
RMSE on 109 predictions (shaded area): 5.284793170023226
```

In [20]:

```
rmses = []
bad_ones = 0
for offset in [YEAR, YEAR+QYEAR, YEAR+2*QYEAR]:
    for evaldata in evaltemps:
        prime_data = evaldata[offset:offset+PRIMELEN]
        results = prediction_run(predict_fn, prime_data, RUNLEN)
        rmse = math.sqrt(np.mean((evaldata[offset+PRIMELEN:offset+PRIMELEN+RMSEL
        rmses.append(rmse)
        if rmse>7: bad_ones += 1
        print("RMSE on {} predictions (shaded area): {}".format(RMSELEN, rmse))
print("Average RMSE on {} weather stations: {} ({} really bad ones, i.e. >7.0)".
sys.stdout.flush()
```

```
RMSE on 109 predictions (shaded area): 5.999025097714327
RMSE on 109 predictions (shaded area): 4.347853590525463
RMSE on 109 predictions (shaded area): 7.298584728826519
RMSE on 109 predictions (shaded area): 6.628857072735798
RMSE on 109 predictions (shaded area): 5.1630839713896854
RMSE on 109 predictions (shaded area): 5.497934147648922
RMSE on 109 predictions (shaded area): 4.818790960426101
RMSE on 109 predictions (shaded area): 6.167821331250168
RMSE on 109 predictions (shaded area): 8.17302636577258
RMSE on 109 predictions (shaded area): 4.325787552690062
RMSE on 109 predictions (shaded area): 12.211649154511946
RMSE on 109 predictions (shaded area): 8.580636020828683
RMSE on 109 predictions (shaded area): 4.24402536272085
RMSE on 109 predictions (shaded area): 3.835527322351085
RMSE on 109 predictions (shaded area): 3.8656372535639156
RMSE on 109 predictions (shaded area): 4.2736031486063055
RMSE on 109 predictions (shaded area): 6.330179434914699
RMSE on 109 predictions (shaded area): 4.9586562187583425
RMSE on 109 predictions (shaded area): 9.380311263025362
RMSE on 109 predictions (shaded area): 8.28450195860041
RMSE on 109 predictions (shaded area): 4.379368241137865
RMSE on 109 predictions (shaded area): 4.100467788086305
RMSE on 109 predictions (shaded area): 5.204311821771394
RMSE on 109 predictions (shaded area): 4.386242881437469
Average RMSE on 8 weather stations: 5.935661778720594 (6 really bad ones, i.e. >
7.0)
```

References

- This notebook is taken from Google's *Tensorflow Without a PhD* series created by Martin Gorner.
- Minor modifications were made. See the source notebooks here