Import Libraries

```
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import DataLoader, Dataset

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import tqdm
import random
import glob
```

Create single input file

This section loads forcings, attributes and discharge data for all the basins and merges the dataframes. The final data is saved as pickle file for easier access to model.

```
root_folder = '/mh1/Atakallou/MLproject'
discharge_files =
glob.glob(f'{root_folder}/basin_dataset_public_v1p2/usgs_streamflow/
*/*.txt')
forcing_files =
glob.glob(f'{root_folder}/basin_dataset_public_v1p2/basin_mean_forcing
/nldas/*/*.txt')

forcings=[]
for f in forcing_files:
    gaugeid = int(f.split('/')[-1].split('_')[0])
    df = pd.read_csv(f,delimiter='\\
s+',skipinitialspace=True,skiprows=3)
    df.insert(0, 'gauge_id', gaugeid)
    forcings.append(df)

forcing = pd.concat(forcings,ignore_index=True)
dates = (forcing.Year.map(str) + "/" + forcing.Mnth.map(str) + "/" +
forcing.Day.map(str))
forcing.insert(0, 'Date', pd.to_datetime(dates, format="%Y/%m/%d"))
forcing.drop(columns=['Year','Mnth','Day','Hr','Dayl(s)'],inplace=True
)

attrs = pd.read_csv(f'{root_folder}/camels_topo.txt',delimiter=';')
for attr in ['camels_clim','camels_vege','camels_geol','camels_soil']:

attrs=pd.merge(attrs,pd.read_csv(f'{root_folder}/{attr}.txt',delimiter
=';'),on='gauge_id')
```

```
attrs.drop(columns=['gauge_lat', 'gauge_lon',
'high_prec_timing','low_prec_timing','dom_land_cover','geol_1st_class'
,

'geol_2nd_class','area_geospa_fabric','root_depth_50','root_depth_99',
'geol_porostiy'],
          inplace=True)

Qobss=[]
for f in discharge_files:
    df = pd.read_csv(f,names =
['gauge_id','year','month','day','Qobs','flag'],delimiter='
',skipinitialspace=True)
    Qobss.append(df)
Qobs = pd.concat(Qobss,ignore_index=True)
dates = (Qobs.year.map(str) + "/" + Qobs.month.map(str) + "/" +
Qobs.day.map(str))
Qobs.insert(0, 'Date', pd.to_datetime(dates, format="%Y/%m/%d"))
Qobs.drop(columns=['year','month','day','flag'],inplace=True)

Qobs = Qobs[Qobs['gauge_id'].isin(attrs['gauge_id'].unique())]

forcing =
forcing[forcing['gauge_id'].isin(attrs['gauge_id'].unique())]

(forcing.merge(attrs)).merge(Qobs).to_pickle('/mh1/nkarki/MLproj/
CAMELS_data.pkl')
```

Dataset preparation

This section creates dataloader for train, valid and test periods

```
root_dir = '/mh1/nkarki/MLproj'

camels_df = pd.read_pickle(f'{root_dir}/CAMELS_data.pkl')

#drop some invalid attributes
camels_df.drop(columns=['dom_land_cover_frac',
 'glim_1st_class_frac',
 'glim_2nd_class_frac',
 'organic_frac',
 'water_frac',
 'other_frac'],inplace=True)

#Remove instances of negative Qobs
camels_df = camels_df[camels_df.Qobs >=
0].sort_values(by=['gauge_id','Date'])

#Load valid basins (n = 531)
basins = pd.read_csv('basins.txt',names=['gauge_id']).gauge_id
```

```
#sample random 480 basins from the pool
train_basins = random.sample(sorted(basins),480)

#Assign remaining as test
test_basins = [basin for basin in basins if basin not in train_basins]

len(test_basins)
```

```
51
```

```
df = camels_df[camels_df.gauge_id.isin(train_basins)]

train_period = pd.date_range('1980-01-01','2008-12-31')
valid_period = pd.date_range('2009-01-01','2014-12-31')
test_period = pd.date_range('1980-01-01','2014-12-31')

train_df = df[df.Date.isin(train_period)]
valid_df = df[df.Date.isin(test_period)]

mean = train_df.mean()
std = train_df.std()

def create_seq(x,y,seq_length):
        num_samples, num_features = x.shape

        x_new = np.zeros((num_samples - seq_length + 1, seq_length,
num_features))
        y_new = np.zeros((num_samples - seq_length + 1, 1))

        for i in range(0, x_new.shape[0]):
            x_new[i, :, :num_features] = x[i:i + seq_length, :]
            y_new[i, :] = y[i + seq_length - 1, 0]

        return x_new.astype('float16'), y_new.astype('float16')

X_train = []
y_train = []
for idx,df in train_df.groupby('gauge_id'):
    df = df.drop(columns=['Date','gauge_id'])
    df = (df - mean[2:])/std[2:]
    X = df.iloc[:,:-1].to_numpy('float32')
    y = df.iloc[:,-1].to_numpy('float32').reshape(-1,1)
    X,y = create_seq(X,y,270)
    X_train.append(X)
    y_train.append(y)

X_valid = []
y_valid = []
for idx,df in valid_df.groupby('gauge_id'):
    df = df.drop(columns=['Date','gauge_id'])
    df = (df - mean[2:])/std[2:]
    X = df.iloc[:,:-1].to_numpy('float32')
```

```
    y = df.iloc[:,-1].to_numpy('float32').reshape(-1,1)
    X,y = create_seq(X,y,270)
    X_valid.append(X)
    y_valid.append(y)

test_df = camels_df[camels_df.gauge_id.isin(test_basins)]

X_test = []
y_test = []
for idx,df in test_df.groupby('gauge_id'):
    df = df.drop(columns=['Date','gauge_id'])
    df = (df - mean[2:])/std[2:]
    X = df.iloc[:,:-1].to_numpy('float32')
    y = df.iloc[:,-1].to_numpy('float32').reshape(-1,1)
    X,y = create_seq(X,y,270)
    X_test.append(X)
    y_test.append(y)

X_train = np.concatenate(X_train)
y_train = np.concatenate(y_train)
X_valid = np.concatenate(X_valid)
y_valid = np.concatenate(y_valid)
X_test = np.concatenate(X_test)
y_test = np.concatenate(y_test)

X_train.shape
```

```
(4855900, 270, 32)
```

```
class CAMELS(Dataset):
    def __init__(self,X,y,qstd):
        self.X = torch.from_numpy(X.copy()).float()
        self.y = torch.from_numpy(y.copy()).float()
        self.qstd = qstd

    def __len__(self):
        return len(self.X)

    def __getitem__(self,idx:int):
        return self.X[idx],self.y[idx],self.qstd[idx]

train_data = CAMELS(X_train,y_train)
valid_data = CAMELS(X_valid,y_valid)
test_data = CAMELS(X_test,y_test)
train_loader = DataLoader(train_data, batch_size=120, shuffle=True)
valid_loader = DataLoader(valid_data, batch_size=120, shuffle=False)
test_loader = DataLoader(test_data,batch_size=2000,shuffle=False)
```

Model

```
class Model(nn.Module):

    def __init__(self, hidden_size, dropout_rate=0.0):

        super(Model, self).__init__()
        self.hidden_size = hidden_size
        self.dropout_rate = dropout_rate

        self.lstm = nn.LSTM(input_size=32,
hidden_size=self.hidden_size,
                            num_layers=1, bias=True, batch_first=True)
        self.dropout = nn.Dropout(p=self.dropout_rate)
        self.fc = nn.Linear(in_features=self.hidden_size,
out_features=1)

    def forward(self, x):
        output, (h_n, c_n) = self.lstm(x)
        # perform prediction only at the end of the input sequence
        pred = self.fc(self.dropout(h_n[-1,:,:]))
        return pred

def train_epoch(model, optimizer, loader, loss_func, epoch):
    model.to(DEVICE)
    model.train()
    pbar = tqdm.tqdm_notebook(loader)
    pbar.set_description(f"Epoch {epoch}")

    for xs, ys in pbar:
        optimizer.zero_grad()
        xs, ys = xs.to(DEVICE), ys.to(DEVICE)
        y_hat = model(xs)
        loss = loss_func(y_hat, ys)
        loss.backward()
        optimizer.step()
        pbar.set_postfix_str(f"Loss: {loss.item():.4f}")


def eval_model(model, loader):
    model.to(DEVICE)
    model.eval()
    obs = []
    preds = []
    with torch.no_grad():
        for xs, ys in loader:
            xs = xs.to(DEVICE)
            y_hat = model(xs)
            obs.append(ys)
            preds.append(y_hat)
```

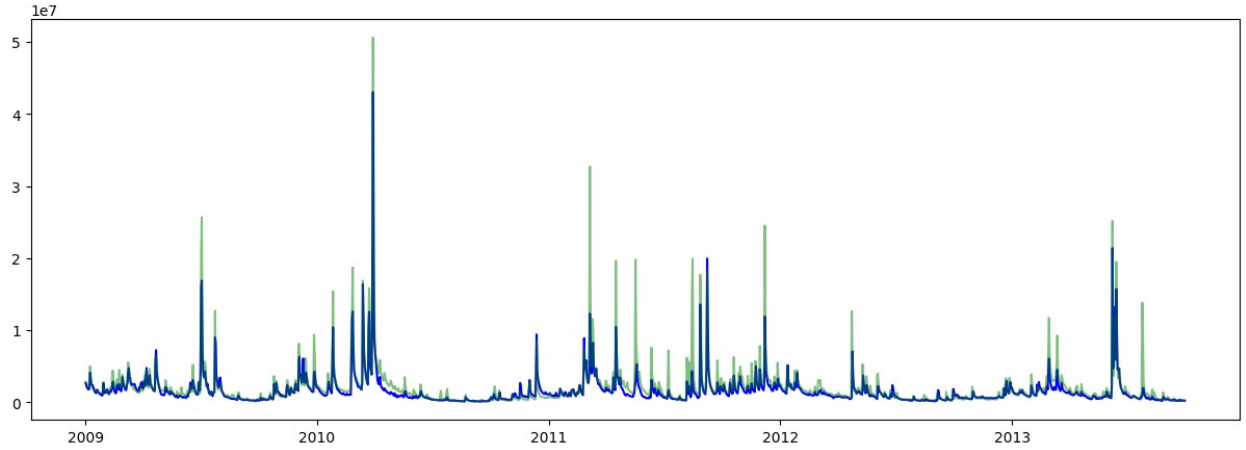
```
    return torch.cat(obs), torch.cat(preds)

DEVICE = torch.device("cuda:0" if torch.cuda.is_available() else
"cpu") # This line checks if GPU is available
DEVICE
```

```
device(type='cuda', index=0)
```

```
model = Model(hidden_size = 128, dropout_rate= 0.4)
optimizer = torch.optim.Adam(model.parameters(),lr=0.001)
criterion = nn.MSELoss()

n_epochs = 100 # Number of training epochs

for i in range(n_epochs):
    train_epoch(model, optimizer, train_loader, criterion, i+1)

obs, preds = eval_model(model, valid_loader)
obs = obs.detach().to('cpu').numpy()
preds = preds.detach().to('cpu').numpy()

#Rescale
obs = (obs*std.iloc[-1])+mean.iloc[-1]
preds = (preds*std.iloc[-1])+mean.iloc[-1]

#Align prediction with the original dataframe
pred_df = valid_df.groupby('gauge_id').tail(-269)
pred_df = pred_df[['Date','gauge_id']]
pred_df['Qobs'] = obs.flatten()
pred_df['Qsim'] = preds.flatten()

pred_df = pred_df[pred_df.Date.isin(valid_period)]

pred_df.to_csv('/scratch/nkarki/lstm_preds_filtered.csv')

pred_df['Date'] = pd.to_datetime(pred_df.Date)

pred_df = pred_df[pred_df.Date.isin(valid_period)]

ids = []
KGE = []
for idx,df in pred_df.groupby('gauge_id'):
    ids.append(idx)
    obs = df["Qobs"].values
    sim = df["Qsim"].values
    sim = sim[obs >= 0]
    obs = obs[obs >= 0]
    kge = calc_kge(obs,sim)
    KGE.append(kge)

valid_KGE = pd.DataFrame({'gauge_id':ids,'KGE':KGE})
```

```
#save validation results to csv
valid_KGE.to_csv('/scratch/nkarki/valid_KGE.csv')

#Plot validation results for a basin
plot_df = pred_df[pred_df.gauge_id == 1123000]
plt.figure(figsize=(15, 5))  # Set the figure size

plt.plot(plot_df.Date, plot_df.Qsim, label='Qsim',color='blue')
plt.plot(plot_df.Date, plot_df.Qobs, label='Qobs',
color='green',alpha=0.5)
```

```
[<matplotlib.lines.Line2D at 0x2b606a537b20>]
```

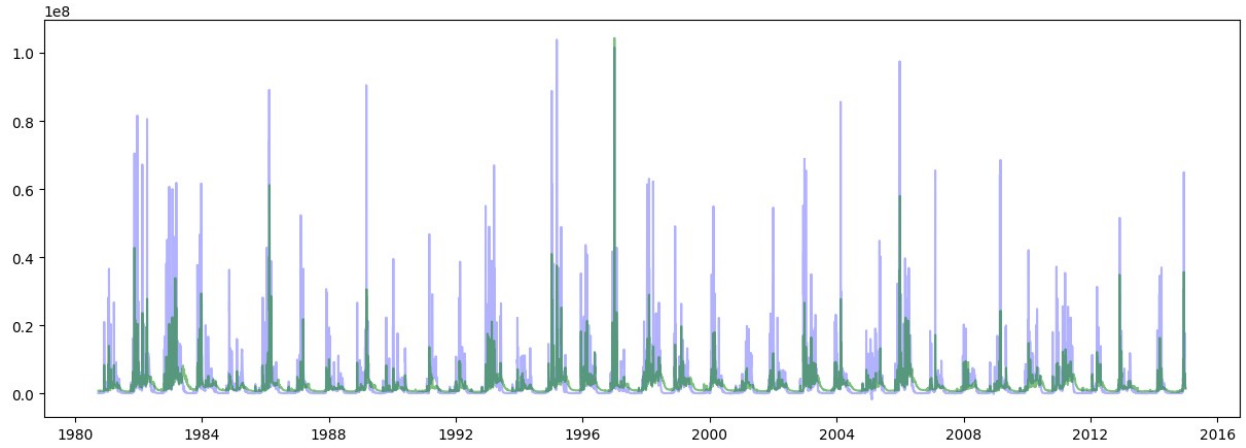


Ungauged basins: Prediction for test set

```
obs, preds = eval_model(model, test_loader)
obs = obs.detach().to('cpu').numpy()
pub_preds = preds.detach().to('cpu').numpy()

obs = (obs*std.iloc[-1])+mean.iloc[-1]
pub_preds = (pub_preds*std.iloc[-1])+mean.iloc[-1]

pred_df = test_df.groupby('gauge_id').tail(-269)
pred_df = pred_df[['Date','gauge_id']]
pred_df['Qobs'] = obs.flatten()
pred_df['Qsim'] = pub_preds.flatten()

#Save test results to csv
pred_df.to_csv('/scratch/nkarki/pub_eval_filtered.csv')

#View results for a basin
plot_df = pred_df[pred_df.gauge_id == 11381500]

plt.figure(figsize=(15, 5))
plt.plot(plot_df.Date, plot_df.Qsim,
label='Qsim',color='blue',alpha=0.3)
```

```
plt.plot(plot_df.Date, plot_df.Qobs, label='Qobs',
color='green',alpha=0.5)
```

```
[<matplotlib.lines.Line2D at 0x2b5fc2b48a30>]
```



Compute KGE and plots for report

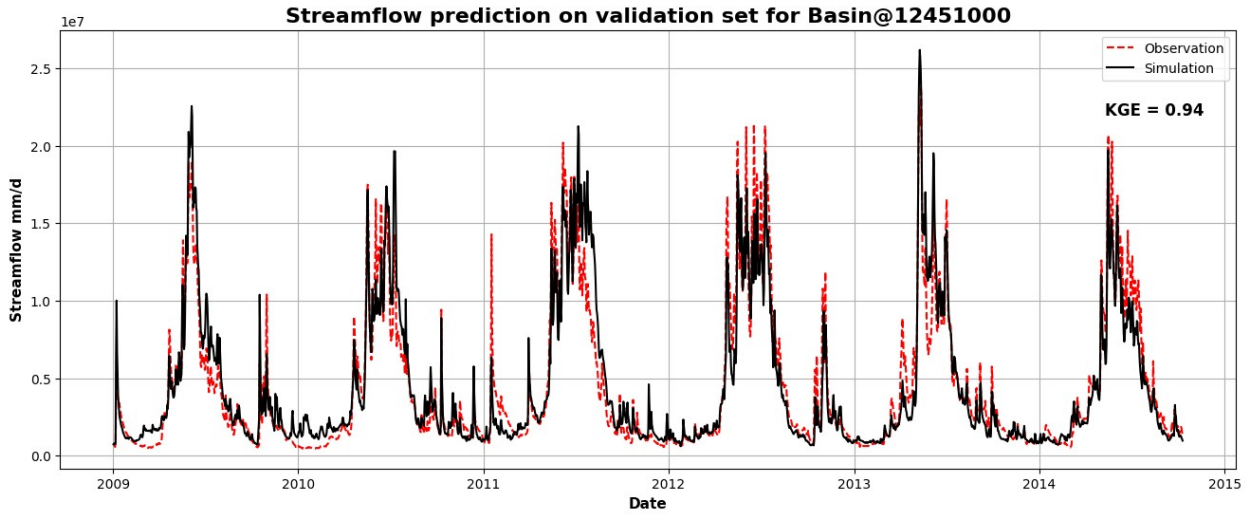
```
#reading the saved outputs for model predictions
pred_df = pd.read_csv('/scratch/nkarki/lstm_preds_filtered.csv')

#defining correlation function which will be used in KGE function
def compute_correlation(observation, prediction):
    prediction = prediction.to_numpy()
    observation = observation.to_numpy()
    obs_mean = np.mean(observation)
    pred_mean = np.mean(prediction)
    covariance = np.mean((observation - obs_mean) * (prediction -
pred_mean))
    obs_std = np.sqrt(np.mean((observation - obs_mean) ** 2))
    pred_std = np.sqrt(np.mean((prediction - pred_mean) ** 2))
    correlation_coefficient = covariance / (obs_std * pred_std)
    return correlation_coefficient.item()
#definig KGE function
def kge(observation, prediction):
    r = compute_correlation(observation, prediction)
    prediction = prediction.to_numpy()
    observation = observation.to_numpy()
    mean_obs = np.mean(observation)
    mean_pred = np.mean(prediction)
    std_obs = np.std(observation)
    std_pred = np.std(prediction)
    alpha = std_pred / std_obs
    beta = mean_pred / mean_obs
    kge = 1 - np.sqrt((alpha - 1)**2 + (beta - 1)**2 + (r - 1)**2)
    return kge.item()
```

```
#compute KGE value for each gauge based on the model output
ids = []
KGE = []
MAX_KGE= 0
split_date = pd.Timestamp('2009-01-01')
for idx,df in pred_df.groupby('gauge_id'):
    df['Date'] = pd.to_datetime(df['Date'])
    df = df[df['Date'] > split_date]
    ids.append(idx)
    gage_kge = kge(df.Qobs,df.Qsim)
    KGE.append(gage_kge)
    if gage_kge > MAX_KGE:
        best_gauge = { 'Date' : df.Date,
                       'best_gage_id': df.gauge_id,
                       'best_Qobs': df.Qobs,
                       'best_Qsim' : df.Qsim}
        best_gauge = pd.DataFrame(best_gauge)
        MAX_KGE = gage_kge
    print(idx,gage_kge)

gauged_KGE= KGE

#Save the KGE outputs for each gauge
KGE_data = { 'gauge_id' : ids, 'KGE' : KGE }
KGE_df= pd.DataFrame (KGE_data)
KGE_df.to_csv('/scratch/nkarki/KGE_preds.csv', index=False)

#get original values for model plot
root_dir = '/scratch/nkarki'
camels_df = pd.read_pickle(f'{root_dir}/CAMELS_data.pkl')
basins = camels_df.gauge_id
camels_df.drop(columns=['dom_land_cover_frac',
 'glim_1st_class_frac',
 'glim_2nd_class_frac',
 'organic_frac',
 'water_frac',
 'other_frac'],inplace=True)
df = camels_df[camels_df.gauge_id.isin(basins)]
test_period = pd.date_range('1980-01-01','2014-12-31')
valid_df = df[df.Date.isin(test_period)]
bestOriginaldata = df[valid_df['gauge_id'] ==
best_gauge['best_gage_id'].iloc[0]]
bestOriginaldata = bestOriginaldata[bestOriginaldata['Date'] >
split_date]

#plot the streamflow prediction for the best gauge
plt.figure(figsize=(16, 6))  # Optional: Sets the figure size
plt.plot(pd.to_datetime(best_gauge['Date']).values,
best_gauge['best_Qobs'].to_numpy(), linestyle='--', color='red',
label='Observation')  # 'marker' parameter is optional, adds markers
```

```
at data points
plt.plot(pd.to_datetime(best_gauge['Date']).values,
best_gauge['best_Qsim'].to_numpy(), color='black', label='Simulation')
# Adding titles and labels
x = best_gauge['best_gage_id'].iloc[0]
plt.title(f'Streamflow prediction on validation set for Basin@{x}',
fontsize=16, fontweight='bold')
plt.xlabel('Date', fontweight='bold', fontname='Times New Roman',
fontsize=11)
plt.ylabel('Streamflow mm/d', fontweight='bold', fontname='Times New
Roman', fontsize=11)

plt.text(x=16200, y=22000000,   # x,y position of start of text
         s=f'KGE = {MAX_KGE:.2f}', fontsize=12, fontweight='bold',
fontname='Times New Roman',  # Text
         bbox=dict(facecolor='red', alpha=0 , ))  # Textbox style
plt.legend()
# Display the plot
plt.savefig('ValidationPlot.png', dpi=300)
plt.grid(True)  # Optional: Adds a grid
plt.show()
```

```
/state/partition1/5292522/ipykernel_35905/18065895.py:16:
MatplotlibDeprecationWarning: savefig() got unexpected keyword
argument "facecolor" which is no longer supported as of 3.3 and will
become an error two minor releases later
  plt.savefig('ValidationPlot.png', dpi=300)
/state/partition1/5292522/ipykernel_35905/18065895.py:16:
MatplotlibDeprecationWarning: savefig() got unexpected keyword
argument "edgecolor" which is no longer supported as of 3.3 and will
become an error two minor releases later
  plt.savefig('ValidationPlot.png', dpi=300)
/state/partition1/5292522/ipykernel_35905/18065895.py:16:
MatplotlibDeprecationWarning: savefig() got unexpected keyword
argument "orientation" which is no longer supported as of 3.3 and will
become an error two minor releases later
  plt.savefig('ValidationPlot.png', dpi=300)
/state/partition1/5292522/ipykernel_35905/18065895.py:16:
MatplotlibDeprecationWarning: savefig() got unexpected keyword
argument "bbox_inches_restore" which is no longer supported as of 3.3
and will become an error two minor releases later
  plt.savefig('ValidationPlot.png', dpi=300)
findfont: Font family ['Times New Roman'] not found. Falling back to
DejaVu Sans.
findfont: Font family ['Times New Roman'] not found. Falling back to
DejaVu Sans.
/bighome/atakallou/.conda/envs/pytorch_env/lib/python3.11/site-
packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning:
savefig() got unexpected keyword argument "orientation" which is no
longer supported as of 3.3 and will become an error two minor releases
```

```
later
  fig.canvas.print_figure(bytes_io, **kw)
/bighome/atakallou/.conda/envs/pytorch_env/lib/python3.11/site-
packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning:
savefig() got unexpected keyword argument "facecolor" which is no
longer supported as of 3.3 and will become an error two minor releases
later
  fig.canvas.print_figure(bytes_io, **kw)
/bighome/atakallou/.conda/envs/pytorch_env/lib/python3.11/site-
packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning:
savefig() got unexpected keyword argument "edgecolor" which is no
longer supported as of 3.3 and will become an error two minor releases
later
  fig.canvas.print_figure(bytes_io, **kw)
/bighome/atakallou/.conda/envs/pytorch_env/lib/python3.11/site-
packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning:
savefig() got unexpected keyword argument "bbox_inches_restore" which
is no longer supported as of 3.3 and will become an error two minor
releases later
  fig.canvas.print_figure(bytes_io, **kw)
```

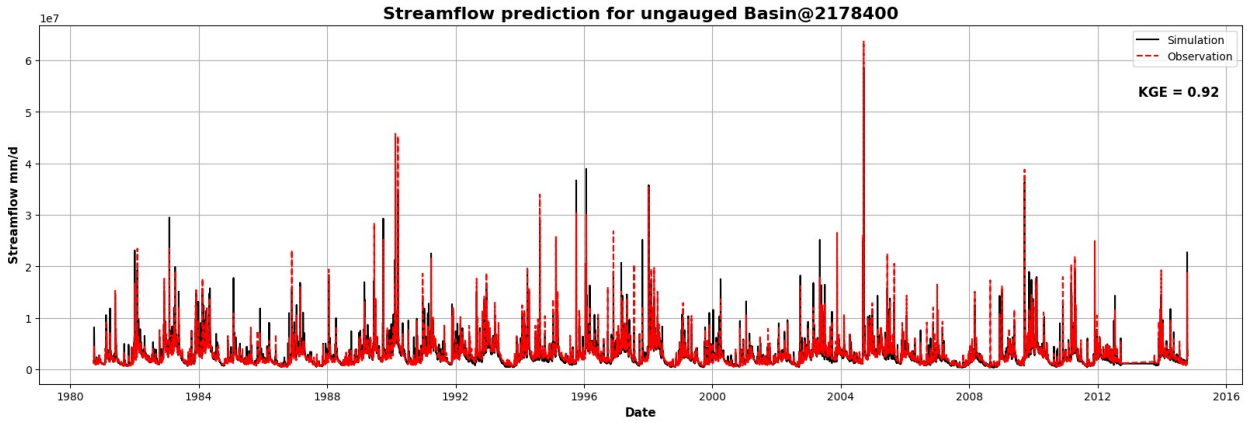



```
#read the data from the ungauged output file and compute KGE for each
gauge
ungauged_df = pd.read_csv('/scratch/nkarki/pub_eval_filtered.csv')
ids = []
pred_KGE = KGE
KGE = []
MAX_KGE= 0
for idx,df in ungauged_df.groupby('gauge_id'):
    ids.append(idx)
    ungauged_kge = kge(df.Qobs,df.Qsim)
    KGE.append(ungauged_kge)
    if ungauged_kge > MAX_KGE:
```

```
        best_ungauged = { 'Date' : df.Date,
                        'best_gage_id': df.gauge_id,
                        'best_Qobs': df.Qobs,
                        'best_Qsim' : df.Qsim}
        best_ungauged = pd.DataFrame(best_ungauged)
        MAX_KGE = ungauged_kge
    print(idx,ungauged_kge)

ungageKGE= KGE

#plot the streamflow prediction for ungauged basin
plt.figure(figsize=(20, 6))  # Optional: Sets the figure size
# plt.plot(pd.to_datetime(bestOriginaldata['Date']).values,
bestOriginaldata['Qobs'].to_numpy())  # 'marker' parameter is
optional, adds markers at data points
plt.plot(pd.to_datetime(best_ungauged['Date']).values,
best_ungauged['best_Qsim'].to_numpy(), color='black',
label='Simulation')
plt.plot(pd.to_datetime(best_ungauged['Date']).values,
best_ungauged['best_Qobs'].to_numpy(), linestyle='--', color='red',
label='Observation')

# Adding titles and labels
x = best_ungauged['best_gage_id'].iloc[0]
plt.title(f'Streamflow prediction for ungauged Basin@{x}',
fontsize=16, fontweight='bold')
plt.xlabel('Date', fontweight='bold', fontname='Times New Roman',
fontsize=11)
plt.ylabel('Streamflow mm/d', fontweight='bold', fontname='Times New
Roman', fontsize=11)

plt.text(x=15800, y=53000000,   # x,y position of start of text
         s=f'KGE = {MAX_KGE:.2f}', fontsize=12, fontweight='bold',
fontname='Times New Roman',  # Text
         bbox=dict(facecolor='red', alpha=0 , ))  # Textbox style

plt.legend()
# Display the plot
plt.savefig('PuBPlot.png', dpi=300)
plt.grid(True)  # Optional: Adds a grid
plt.show()
```

```
/state/partition1/5292522/ipykernel_35905/1783908706.py:19:
MatplotlibDeprecationWarning: savefig() got unexpected keyword
argument "facecolor" which is no longer supported as of 3.3 and will
become an error two minor releases later
  plt.savefig('PuBPlot.png', dpi=300)
/state/partition1/5292522/ipykernel_35905/1783908706.py:19:
MatplotlibDeprecationWarning: savefig() got unexpected keyword
argument "edgecolor" which is no longer supported as of 3.3 and will
```

```
become an error two minor releases later
  plt.savefig('PuBPlot.png', dpi=300)
/state/partition1/5292522/ipykernel_35905/1783908706.py:19:
MatplotlibDeprecationWarning: savefig() got unexpected keyword
argument "orientation" which is no longer supported as of 3.3 and will
become an error two minor releases later
  plt.savefig('PuBPlot.png', dpi=300)
/state/partition1/5292522/ipykernel_35905/1783908706.py:19:
MatplotlibDeprecationWarning: savefig() got unexpected keyword
argument "bbox_inches_restore" which is no longer supported as of 3.3
and will become an error two minor releases later
  plt.savefig('PuBPlot.png', dpi=300)
/bighome/atakallou/.conda/envs/pytorch_env/lib/python3.11/site-
packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning:
savefig() got unexpected keyword argument "orientation" which is no
longer supported as of 3.3 and will become an error two minor releases
later
  fig.canvas.print_figure(bytes_io, **kw)
/bighome/atakallou/.conda/envs/pytorch_env/lib/python3.11/site-
packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning:
savefig() got unexpected keyword argument "facecolor" which is no
longer supported as of 3.3 and will become an error two minor releases
later
  fig.canvas.print_figure(bytes_io, **kw)
/bighome/atakallou/.conda/envs/pytorch_env/lib/python3.11/site-
packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning:
savefig() got unexpected keyword argument "edgecolor" which is no
longer supported as of 3.3 and will become an error two minor releases
later
  fig.canvas.print_figure(bytes_io, **kw)
/bighome/atakallou/.conda/envs/pytorch_env/lib/python3.11/site-
packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning:
savefig() got unexpected keyword argument "bbox_inches_restore" which
is no longer supported as of 3.3 and will become an error two minor
releases later
  fig.canvas.print_figure(bytes_io, **kw)
```

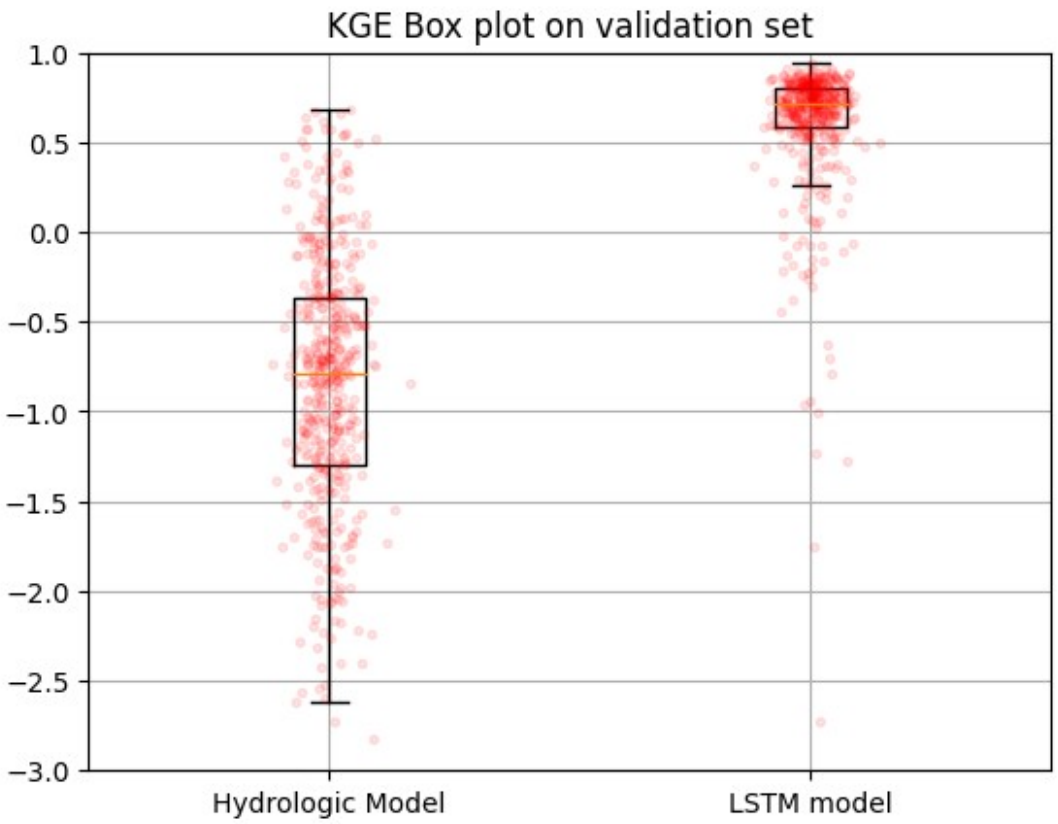


```
import os

#reading the hydrologic model outputs and save them in a csv file this
is optional after first run and you can skip it in the other runs
base_path =
'/mh1/Atakallou/MLproject/model_output_nldas/model_output/flow_timeser
ies/nldas'
end_date = '2014-12-31'

file_pattern = os.path.join(base_path, '**', '*_model_output.txt')
df_list = []
for file_path in glob.glob(file_pattern):
    # Extract gauge_id and subdir from the file path

    parts = file_path.split(os.sep)
    gauge_id = parts[-1].split('_')[0]
    gauge_id = gauge_id.lstrip('0')
    temp_df = pd.read_csv(file_path, delim_whitespace=True,
names=["YR", "MNTH", "DY", "HR", "SWE", "PRCP", "RAIM", "TAIR", "PET",
"ET", "MOD_RUN", "OBS_RUN"])
    temp_df['gauge_id'] = gauge_id
    temp_df = temp_df.iloc[1:]
    date_range = pd.date_range(end=end_date, periods=len(temp_df),
freq='D')
    temp_df['date']=date_range
    temp_df.drop(columns=['SWE',
     'PRCP',
     'RAIM',
     'TAIR',
     'PET',
     'ET',
      'YR',
     'MNTH',
     'DY',
     'HR'],inplace=True)
    df_list.append(temp_df)

full_df = pd.concat(df_list, ignore_index=True)
full_df.to_csv('/mh1/Atakallou/MLproject/final/modelOutput.csv')

    # Read the data from the text file
    # temp_df = pd.read_csv(file_path, delim_whitespace=True,
names=["YR", "MNTH", "DY", "HR", "SWE", "PRCP", "RAIM", "TAIR", "PET",
"ET", "MOD_RUN", "OBS_RUN"])

#read  hydrologic model outputs
ModelOutput_df =
pd.read_csv('/mh1/Atakallou/MLproject/final/modelOutput.csv')
```

```
#filter the hydrologic model outputs based on the gauges used for
model prediction in training part
filtered_df1 =
ModelOutput_df[ModelOutput_df['gauge_id'].isin(pred_df['gauge_id'])]
full_df = filtered_df1[filtered_df1['date'].isin(pred_df['Date'])]
full_df = full_df[full_df['gauge_id'].isin(pred_df['gauge_id'])]
full_df = full_df.drop(columns=['Unnamed: 0'])

#compute KGE value for hydrologic model outputs
ids = []
KGE = []
MAX_KGE= 0
full_df= full_df.dropna()
full_df['OBS_RUN'] = pd.to_numeric(full_df['OBS_RUN'],
errors='coerce')
full_df = full_df.dropna(subset=['OBS_RUN'])
full_df['MOD_RUN'] = pd.to_numeric(full_df['MOD_RUN'],
errors='coerce')
full_df = full_df.dropna(subset=['MOD_RUN'])
for idx,df in full_df.groupby('gauge_id'):
    ids.append(idx)
    print(df.OBS_RUN)
    model_kge = kge(df.OBS_RUN,df.MOD_RUN)
    KGE.append(model_kge)
    if model_kge > MAX_KGE:
        best_model = { 'Date' : df.date,
                      'best_gage_id': df.gauge_id,
                      'best_Qobs': df.OBS_RUN,
                      'best_Qsim' : df.MOD_RUN}
        best_model = pd.DataFrame(best_model)
        MAX_KGE = model_kge
    print(idx,model_kge)

model_validation_kge=KGE

#create box plot to compare the LSTM model and hydrologic models for
gauged basins on validation set
# plt.figure(figsize=(10, 6))
import seaborn as sns

KGE = [x for x in KGE if not np.isnan(x)]
# Create a box plot

plt.boxplot([model_validation_kge, gauged_KGE], showfliers=False)
for i, data in enumerate([model_validation_kge, gauged_KGE], 1):
    y = data
    x = np.random.normal(i, 0.04, size=len(y))
    plt.plot(x, y, 'r.', alpha=0.1)
plt.ylim(-3, 1)
plt.xticks([1, 2], ['Hydrologic Model', 'LSTM model'])
```

```
plt.grid(True)  # Optional: Adds a grid
plt.title('KGE Box plot on validation set')
plt.savefig('ValidationBoxPlot.png', dpi=300)

plt.show()
```

```
/state/partition1/5292522/ipykernel_35905/3345600631.py:17:
MatplotlibDeprecationWarning: savefig() got unexpected keyword
argument "facecolor" which is no longer supported as of 3.3 and will
become an error two minor releases later
  plt.savefig('ValidationBoxPlot.png', dpi=300)
/state/partition1/5292522/ipykernel_35905/3345600631.py:17:
MatplotlibDeprecationWarning: savefig() got unexpected keyword
argument "edgecolor" which is no longer supported as of 3.3 and will
become an error two minor releases later
  plt.savefig('ValidationBoxPlot.png', dpi=300)
/state/partition1/5292522/ipykernel_35905/3345600631.py:17:
MatplotlibDeprecationWarning: savefig() got unexpected keyword
argument "orientation" which is no longer supported as of 3.3 and will
become an error two minor releases later
  plt.savefig('ValidationBoxPlot.png', dpi=300)
/state/partition1/5292522/ipykernel_35905/3345600631.py:17:
MatplotlibDeprecationWarning: savefig() got unexpected keyword
argument "bbox_inches_restore" which is no longer supported as of 3.3
and will become an error two minor releases later
  plt.savefig('ValidationBoxPlot.png', dpi=300)
/bighome/atakallou/.conda/envs/pytorch_env/lib/python3.11/site-
packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning:
savefig() got unexpected keyword argument "orientation" which is no
longer supported as of 3.3 and will become an error two minor releases
later
  fig.canvas.print_figure(bytes_io, **kw)
/bighome/atakallou/.conda/envs/pytorch_env/lib/python3.11/site-
packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning:
savefig() got unexpected keyword argument "facecolor" which is no
longer supported as of 3.3 and will become an error two minor releases
later
  fig.canvas.print_figure(bytes_io, **kw)
/bighome/atakallou/.conda/envs/pytorch_env/lib/python3.11/site-
packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning:
savefig() got unexpected keyword argument "edgecolor" which is no
longer supported as of 3.3 and will become an error two minor releases
later
  fig.canvas.print_figure(bytes_io, **kw)
/bighome/atakallou/.conda/envs/pytorch_env/lib/python3.11/site-
packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning:
savefig() got unexpected keyword argument "bbox_inches_restore" which
is no longer supported as of 3.3 and will become an error two minor
releases later
  fig.canvas.print_figure(bytes_io, **kw)
```




```
filtered_df1 =
ModelOutput_df[ModelOutput_df['gauge_id'].isin(ungauged_df['gauge_id']
)]
filtered_df1 =
filtered_df1[filtered_df1['date'].isin(ungauged_df['Date'])]

full_df = filtered_df1

ids = []
KGE = []
MAX_KGE= 0
full_df= full_df.dropna()
full_df['OBS_RUN'] = pd.to_numeric(full_df['OBS_RUN'],
errors='coerce')
full_df = full_df.dropna(subset=['OBS_RUN'])
full_df['MOD_RUN'] = pd.to_numeric(full_df['MOD_RUN'],
errors='coerce')
full_df = full_df.dropna(subset=['MOD_RUN'])
for idx,df in full_df.groupby('gauge_id'):
    ids.append(idx)
    print(df.OBS_RUN)
    model_kge = kge(df.OBS_RUN,df.MOD_RUN)
    KGE.append(model_kge)
    if model_kge > MAX_KGE:
```

```
        best_model = { 'Date' : df.date,
                       'best_gage_id': df.gauge_id,
                       'best_Qobs': df.OBS_RUN,
                       'best_Qsim' : df.MOD_RUN}
        best_model = pd.DataFrame(best_model)
        MAX_KGE = model_kge
    print(idx,model_kge)
model_test_kge = KGE

#create box plot to compare the LSTM model and hydrologic models for
ungauged basins on test set
# Create a box plot
plt.boxplot([model_test_kge, ungageKGE], showfliers=False)
for i, data in enumerate([model_test_kge, ungageKGE], 1):
    y = data
    x = np.random.normal(i, 0.04, size=len(y))
    plt.plot(x, y, 'r.', alpha=0.3)
plt.ylim(0, 1)
plt.xticks([1, 2], ['Hydrologic Model', 'LSTM model'])
plt.grid(True)  # Optional: Adds a grid
plt.title('KGE Box plot for Ungauged Basins')
plt.savefig('BoxPlot_ungauged.png', dpi=300)

plt.show()
```

```
/state/partition1/5292522/ipykernel_35905/355901812.py:12:
MatplotlibDeprecationWarning: savefig() got unexpected keyword
argument "facecolor" which is no longer supported as of 3.3 and will
become an error two minor releases later
  plt.savefig('BoxPlot_ungauged.png', dpi=300)
/state/partition1/5292522/ipykernel_35905/355901812.py:12:
MatplotlibDeprecationWarning: savefig() got unexpected keyword
argument "edgecolor" which is no longer supported as of 3.3 and will
become an error two minor releases later
  plt.savefig('BoxPlot_ungauged.png', dpi=300)
/state/partition1/5292522/ipykernel_35905/355901812.py:12:
MatplotlibDeprecationWarning: savefig() got unexpected keyword
argument "orientation" which is no longer supported as of 3.3 and will
become an error two minor releases later
  plt.savefig('BoxPlot_ungauged.png', dpi=300)
/state/partition1/5292522/ipykernel_35905/355901812.py:12:
MatplotlibDeprecationWarning: savefig() got unexpected keyword
argument "bbox_inches_restore" which is no longer supported as of 3.3
and will become an error two minor releases later
  plt.savefig('BoxPlot_ungauged.png', dpi=300)
/bighome/atakallou/.conda/envs/pytorch_env/lib/python3.11/site-
packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning:
savefig() got unexpected keyword argument "orientation" which is no
longer supported as of 3.3 and will become an error two minor releases
later
```

```
  fig.canvas.print_figure(bytes_io, **kw)
/bighome/atakallou/.conda/envs/pytorch_env/lib/python3.11/site-
packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning:
savefig() got unexpected keyword argument "facecolor" which is no
longer supported as of 3.3 and will become an error two minor releases
later
  fig.canvas.print_figure(bytes_io, **kw)
/bighome/atakallou/.conda/envs/pytorch_env/lib/python3.11/site-
packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning:
savefig() got unexpected keyword argument "edgecolor" which is no
longer supported as of 3.3 and will become an error two minor releases
later
  fig.canvas.print_figure(bytes_io, **kw)
/bighome/atakallou/.conda/envs/pytorch_env/lib/python3.11/site-
packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning:
savefig() got unexpected keyword argument "bbox_inches_restore" which
is no longer supported as of 3.3 and will become an error two minor
releases later
  fig.canvas.print_figure(bytes_io, **kw)
```