HW 7
Machine Learning
PCA-RNN

HW 7 Problem Statement (provided by Dr. Han Hu, MEEG 491V/591V-028, Fall 2021)
Problem 7-1 Pincipal Component Analysis and Recurrent Neural Networks:

The images under ocean/projects/mch210006p/shared/HW7 represent a boiling image sequence under transient heat loads. The images have a frame rate of 1,000 fps (or a time step of 1 ms). Run SVD or PCA to obtain the PC profiles of the image sequence. Feed the extracted PC profiles to an RNN model to forecast the PCs of future frames. Reconstruct image sequences using the predicted PCs and compare the reconstructed images against the true images. The recommended RNN models include LSTM or BiLSTM.

Code Running Instructions
The environment is default.

Coding explanation
(Data Loading) The very first step is to load all the images data and reshape it as 128*128. I reshaped images as 240*240 at very first, but when I got PCs of all images and put them to RNN and got the prediction of PCs, then reconstructed them, I got very obscure images. In order to make the images clearly, more PCs should be selected of each image. But 100 PCs of each image are selected which is a heavily load task for my laptop. Then, I chose a smaller reshape size of each image, which is another way to make the original image and predicted image similar. I save the preprocessed data as a readable file so that I can use it at any time without repeating calculation.

(Coding Logic) Some of steps is straight-forward which is similar to what we did in homework 3 and 5. For the whole coding logic, I got the first 100 important PCs of each image and save them in the matrix. Then I used the "getXY_pcs" function to get the overlapped data sequence, which I use PCs of images 0,1,..9 to predict PCs of images 1,2,…,10 and so on so forth. The reason why I chose the length of sequence as 10 is that from the testing result of different length of sequence, the shorter of sequence the better of the prediction. In order to get better prediction, when I constructed the RNN model, I used one layer of LSTM, then I copied the result as 10 times (the length of data sequence is 10) and add another LSTM layer. The result is much better than when I only used one LSTM layer. The batch size is 32, and 80% of training data is used to do the validation, 90% of the whole data is used to training the model, and 10% of the whole dataset is used to test. Early stopping is used to avoid overfitting.

(Evaluation) The validation loss and the reconstructed predicted images is used as evaluation.

Results
The LSTM structure and the training and validation loss is shown below:

```
Epoch 1/100
225/225 - 5s - loss: 2.0788 - val_loss: 2.0781
Epoch 2/100
225/225 - 2s - loss: 2.0470 - val_loss: 2.0505
Epoch 3/100
225/225 - 2s - loss: 2.0168 - val_loss: 2.0321
Epoch 4/100
```

```
225/225 - 2s - loss: 1.9967 - val_loss: 2.0205
Epoch 5/100
225/225 - 2s - loss: 1.9817 - val_loss: 2.0098
Epoch 6/100
225/225 - 2s - loss: 1.9673 - val_loss: 2.0008
Epoch 7/100
225/225 - 2s - loss: 1.9526 - val_loss: 1.9907
Epoch 8/100
225/225 - 2s - loss: 1.9386 - val_loss: 1.9833
Epoch 9/100
225/225 - 2s - loss: 1.9258 - val_loss: 1.9764
Epoch 10/100
225/225 - 2s - loss: 1.9152 - val_loss: 1.9716
Epoch 11/100
225/225 - 3s - loss: 1.9039 - val_loss: 1.9647
Epoch 12/100
225/225 - 3s - loss: 1.8932 - val_loss: 1.9623
Epoch 13/100
225/225 - 3s - loss: 1.8834 - val_loss: 1.9585
Epoch 14/100
225/225 - 2s - loss: 1.8750 - val_loss: 1.9561
Epoch 15/100
225/225 - 2s - loss: 1.8668 - val_loss: 1.9523
Epoch 16/100
225/225 - 3s - loss: 1.8568 - val_loss: 1.9493
Epoch 17/100
225/225 - 3s - loss: 1.8498 - val_loss: 1.9457
Epoch 18/100
225/225 - 2s - loss: 1.8428 - val_loss: 1.9489
Epoch 19/100
225/225 - 3s - loss: 1.8351 - val_loss: 1.9457
Epoch 20/100
225/225 - 2s - loss: 1.8287 - val_loss: 1.9478
Epoch 21/100
225/225 - 2s - loss: 1.8209 - val_loss: 1.9426
Epoch 22/100
225/225 - 2s - loss: 1.8148 - val_loss: 1.9455
Epoch 23/100
225/225 - 2s - loss: 1.8095 - val_loss: 1.9426
Epoch 24/100
225/225 - 3s - loss: 1.8012 - val_loss: 1.9434
Epoch 25/100
225/225 - 3s - loss: 1.7980 - val_loss: 1.9406
Epoch 26/100
225/225 - 3s - loss: 1.7892 - val_loss: 1.9430
Epoch 27/100
225/225 - 3s - loss: 1.7843 - val_loss: 1.9451
Epoch 28/100
225/225 - 3s - loss: 1.7797 - val_loss: 1.9445
Epoch 29/100
225/225 - 2s - loss: 1.7725 - val_loss: 1.9463
Epoch 30/100
225/225 - 2s - loss: 1.7672 - val_loss: 1.9444
Restoring model weights from the end of the best epoch.
```

```
Epoch 00030: early stopping
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 128)               117248
_________________________________________________________________
repeat_vector (RepeatVector) (None, 10, 128)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 10, 128)           131584
_________________________________________________________________
time_distributed (TimeDistri (None, 10, 100)           12900
=================================================================
Total params: 261,732
Trainable params: 261,732
Non-trainable params: 0
_________________________________________________________________
None
```

The comparison of reconstructed first 4 images using the predicted PCs and the true images is shown in Fig 1.

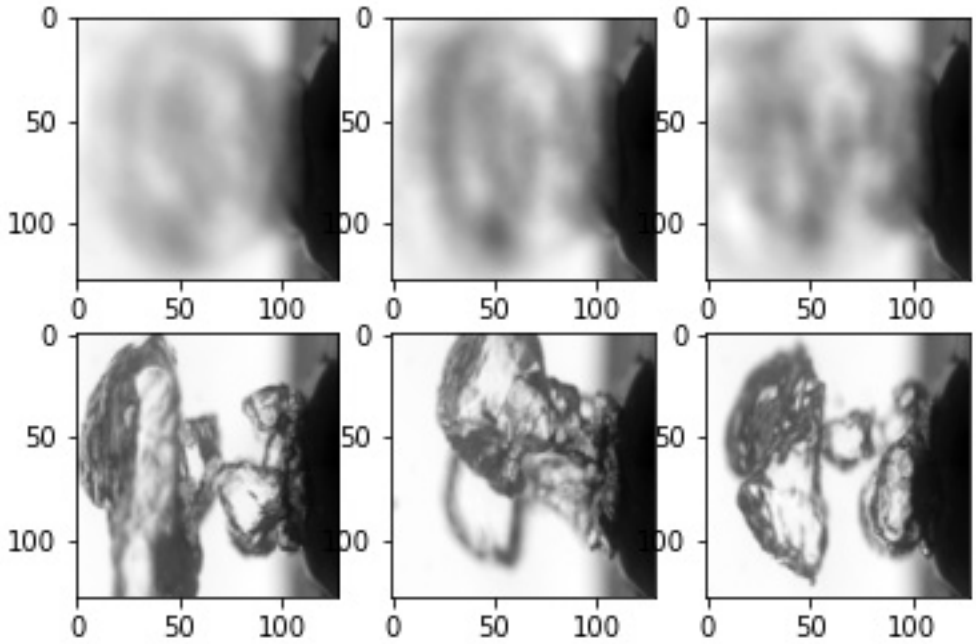

Fig 1. The comparison of reconstructed images using the predicted PCs against the true images.

Challenges

The main challenge is that I tried to use several ways to make the predicted reconstruction images clearly but didn't get any good feedbacks. The training and validation loss get smaller when I changed the reshape size of images, batch size, number of hidden layers, but the predicted reconstruction images don't improve

a lot. Since the time limitation, I haven't tried the other idea that get the prediction sequence by images first then get the 100 pcs of each sequence, then feed the RNN model. This method will have bigger calculation complexity than the one I used in this project, but may get better reconstructed predication images.

References

[1] https://www.tensorflow.org/tutorials/structured_data/time_series#recurrent_neural_network.