上节课回顾

Recurrent Neural Networks

Often data arise as sequences:

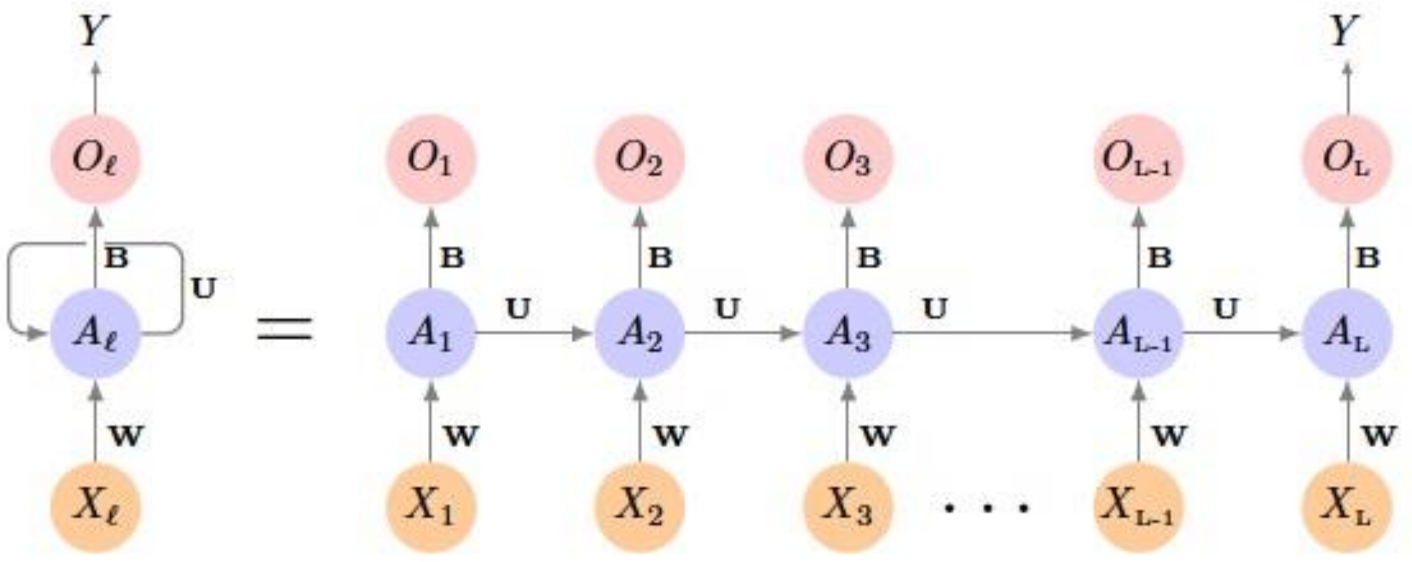
- Documents are sequences of words, and their relative positions have meaning.
- Time-series such as weather data or financial indices.
- Recorded speech or music.
- Handwriting, such as doctor's notes.

RNNs build models that take into account this sequential nature of the data, and build a memory of the past.

- The feature for each observation is a *sequence* of vectors $X = \{X_1, X_2, \ldots, X_L\}$.
- The target Y is often of the usual kind — e.g. a single variable such as `Sentiment`, or a one-hot vector for multiclass.
- However, Y can also be a sequence, such as the same document in a different language.

上节课回顾

Simple Recurrent Neural Network Architecture



- The hidden layer is a sequence of vectors A_ℓ, receiving as input X_ℓ as well as $A_{\ell-1}$. A_ℓ produces an output O_ℓ.
- The *same* weights **W**, **U** and **B** are used at each step in the sequence — hence the term *recurrent*.
- The A_ℓ sequence represents an evolving model for the response that is updated as each element X_ℓ is processed.

上节课回顾

When to Use Deep Learning

- CNNs have had enormous successes in image classification and modeling, and are starting to be used in medical diagnosis. Examples include digital mammography, ophthalmology, MRI scans, and digital X-rays.
- RNNs have had big wins in speech modeling, language translation, and forecasting.

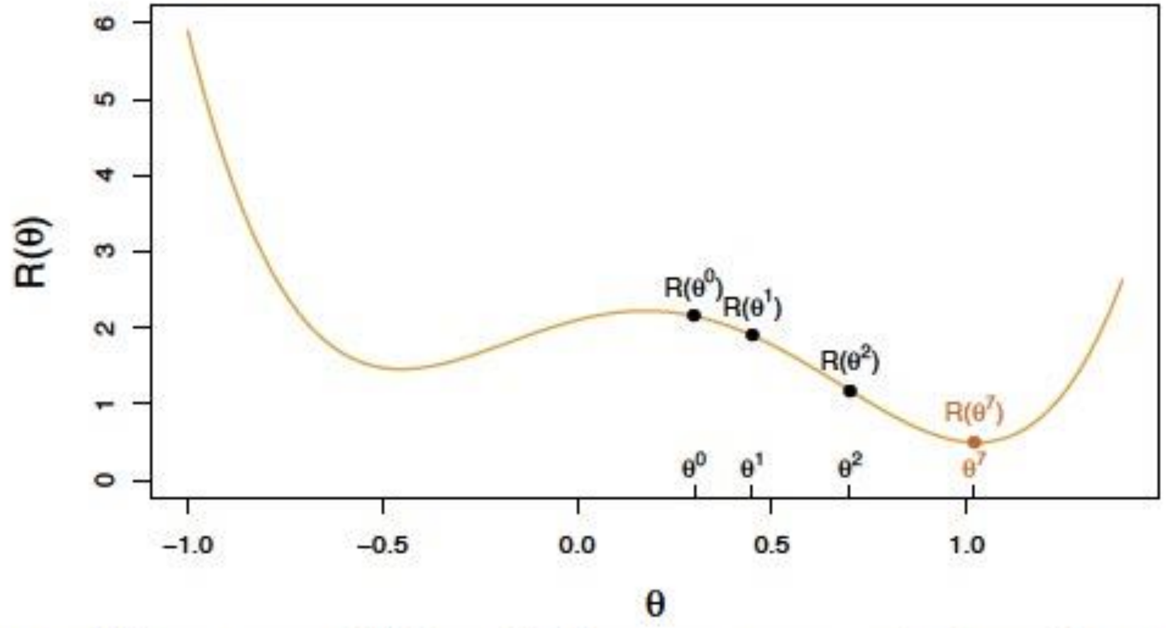
Should we always use deep learning models?

- Often the big successes occur when the *signal to noise ratio* is high — e.g. image recognition and language translation. Datasets are large, and overfitting is not a big problem.
- For noisier data, simpler models can often work better.
    - On the `NYSE` data, the AR(5) model is much simpler than a RNN, and performed as well.
    - On the `IMDB` review data, the linear model fit by `glmnet` did as well as the neural network, and better than the RNN.
- We endorse the *Occam's razor* principal — we prefer simpler models if they work as well. More interpretable!

上节课回顾

Non Convex Functions and Gradient Descent

Let $R(\theta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - f_\theta(x_i))^2$ with $\theta = (\{w_k\}_1^K,\ \beta)$.



1. Start with a guess θ^0 for all the parameters in θ, and set $t = 0$.
2. Iterate until the objective $R(\theta)$ fails to decrease:
   (a) Find a vector δ that reflects a small change in θ, such that $\theta^{t+1} = \theta^t + \delta$ *reduces* the objective; i.e. $R(\theta^{t+1}) < R(\theta^t)$.
   (b) Set $t \leftarrow t + 1$.

上节课回顾

Gradients and Backpropagation

$R(\theta) = \sum_{i=1}^{n} R_i(\theta)$ is a sum, so gradient is sum of gradients.

$$R_i(\theta) = \frac{1}{2}(y_i - f_\theta(x_i))^2 = \frac{1}{2}\Big(y_i - \beta_0 - \sum_{k=1}^{K} \beta_k g\big(w_{k0} + \sum_{j=1}^{p} w_{kj}x_{ij}\big)\Big)^2$$

For ease of notation, let $z_{ik} = w_{k0} + \sum_{j=1}^{p} w_{kj}x_{ij}$.

Backpropagation uses the *chain rule for differentiation*:

$$\begin{aligned}
\frac{\partial R_i(\theta)}{\partial \beta_k} &= \frac{\partial R_i(\theta)}{\partial f_\theta(x_i)} \cdot \frac{\partial f_\theta(x_i)}{\partial \beta_k} \\
&= -(y_i - f_\theta(x_i)) \cdot g(z_{ik}). \\
\frac{\partial R_i(\theta)}{\partial w_{kj}} &= \frac{\partial R_i(\theta)}{\partial f_\theta(x_i)} \cdot \frac{\partial f_\theta(x_i)}{\partial g(z_{ik})} \cdot \frac{\partial g(z_{ik})}{\partial z_{ik}} \cdot \frac{\partial z_{ik}}{\partial w_{kj}} \\
&= -(y_i - f_\theta(x_i)) \cdot \beta_k \cdot g'(z_{ik}) \cdot x_{ij}.
\end{aligned}$$

上节课回顾

Tricks of the Trade

- *Slow learning*. Gradient descent is slow, and a small learning rate ρ slows it even further. With *early stopping*, this is a form of regularization.
- *Stochastic gradient descent*. Rather than compute the gradient using *all* the data, use a small *minibatch* drawn at random at each step. E.g. for `MNIST` data, with $n = 60K$, we use minibatches of 128 observations.
- An *epoch* is a count of iterations and amounts to the number of minibatch updates such that n samples in total have been processed; i.e. $60K/128 \approx 469$ for `MNIST`.
- *Regularization*. Ridge and lasso regularization can be used to shrink the weights at each layer. Two other popular forms of regularization are *dropout* and *augmentation*, discussed next.

上节课回顾

Double Descent

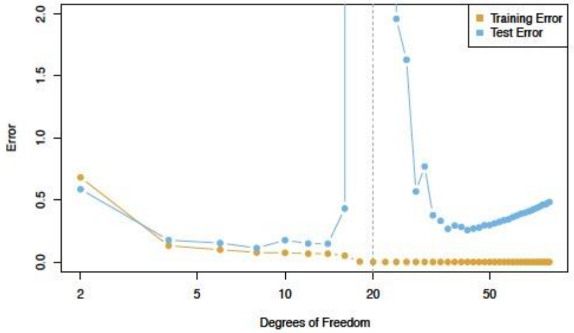
- With neural networks, it seems better to have too many hidden units than too few.
- Likewise more hidden layers better than few.
- Running stochastic gradient descent till zero training error often gives good out-of-sample error.
- Increasing the number of units or layers and again training till zero error sometimes gives *even better* out-of-sample error.

What happened to overfitting and the usual bias-variance trade-off?

Belkin, Hsu, Ma and Mandal (arXiv 2018) *Reconciling Modern Machine Learning and the Bias-Variance Trade-off.*

64 of 72

The Double-Descent Error Curve



- When $d \leq 20$, model is OLS, and we see usual bias-variance trade-off
- When $d > 20$, we revert to minimum-norm. As d increases above 20, $\sum_{j=1}^{d} \hat{\beta}_j^2$ *decreases* since it is easier to achieve zero error, and hence less wiggly solutions.

66 of 72

scientific reports

Check for updates

OPEN

Monthly climate prediction using deep convolutional neural network and long short-term memory

Qingchun Guo[1,2,3,4], Zhenfang He[1,2] & Zhaosheng Wang[5]

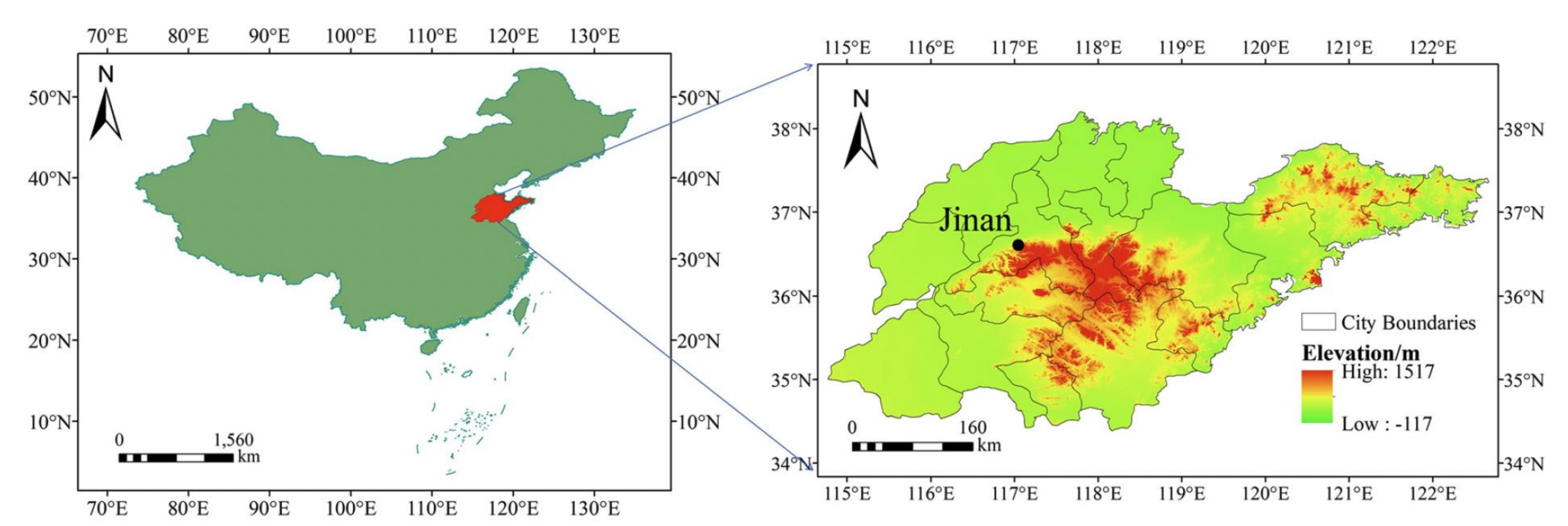


Figure 1. Location Map of the research area. The map is generated using ArcGIS Pro 2.5 (ArcGIS Pro), URL: https://www.esriuk.com.

Goal: 5 machine learning models are used to forecast six climatic factors on a monthly ahead.

Data: The climate data for 72 years (1 January 1951–31 December 2022) used in this study include

1. monthly average atmospheric temperature,
2. extreme minimum atmospheric temperature,
3. extreme maximum atmospheric temperature,
4. precipitation,
5. average relative humidity,
6. sunlight hours.

1

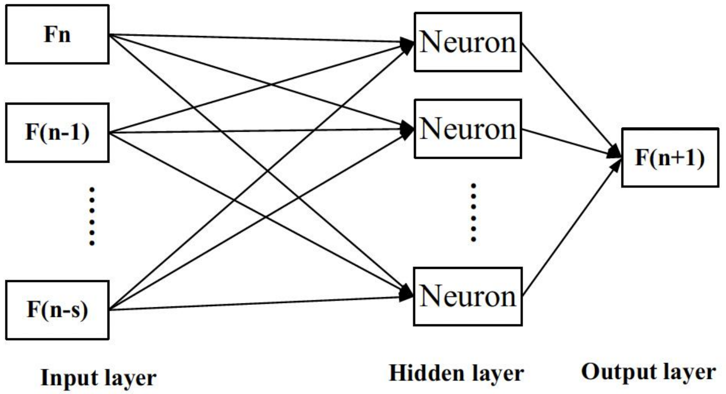


Figure 2. Structure of ANN model.

3

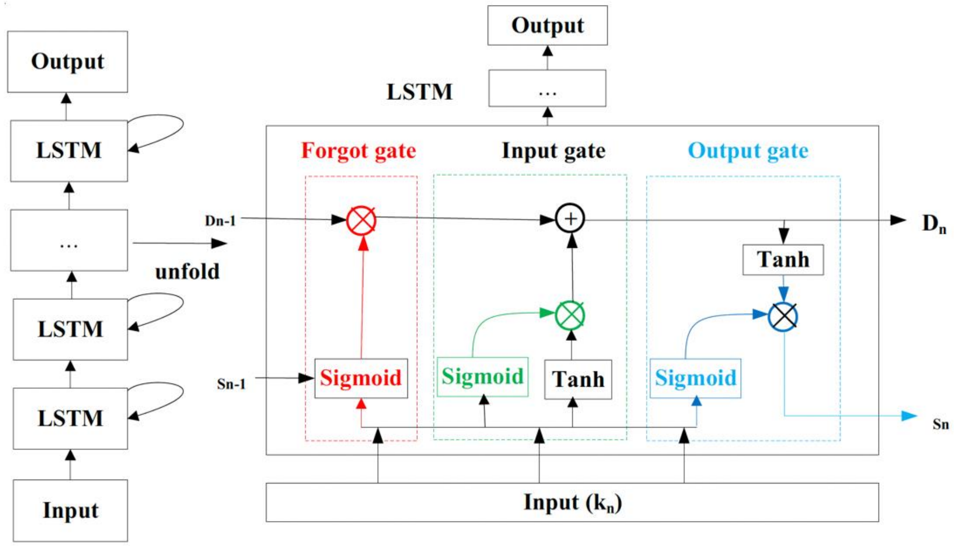


Figure 3. The cell logic structure of the LSTM network.

5

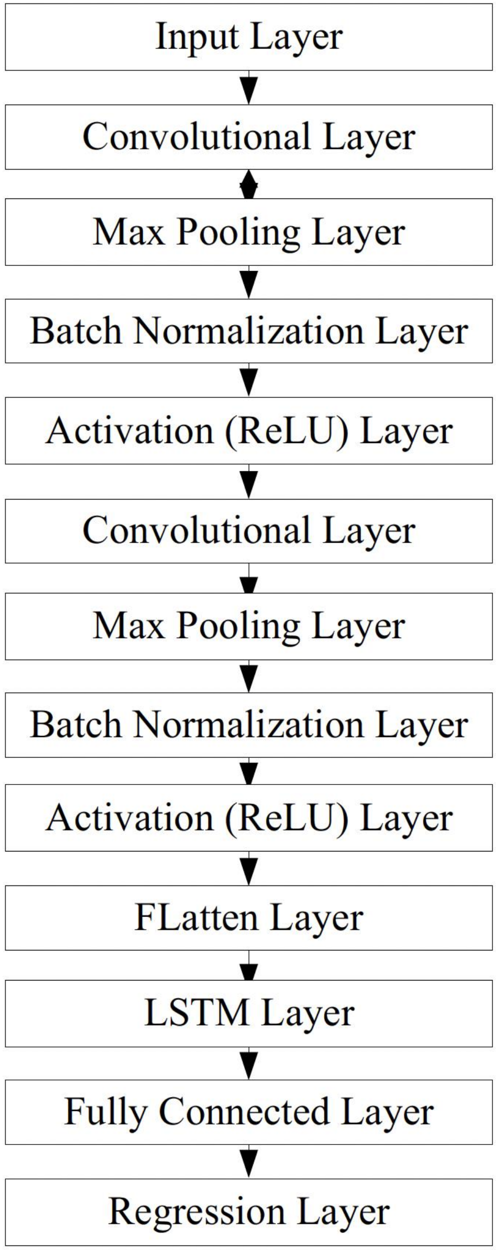


Figure 5. The CNN-LSTM architecture.

2

RNN

4

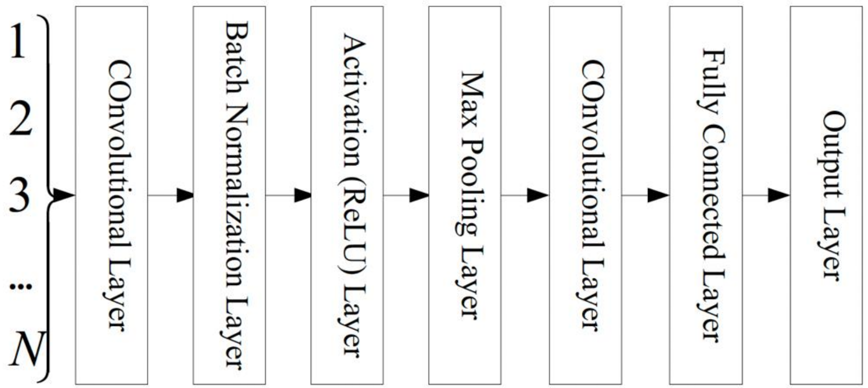


Figure 4. Basic structure diagram of CNN.

Models	R			RMSE (°C)			MAE (°C)		
	Training	Verification	Predicting	Training	Verification	Predicting	Training	Verification	Predicting
ANN	0.9894	0.9870	0.9907	1.8199	1.9923	2.0669	1.4787	1.6352	1.8042
RNN	0.9905	0.9881	0.9895	1.3891	1.5245	1.4416	1.0836	1.1594	1.1917
LSTM	0.9906	0.9870	0.9914	1.3819	1.5965	1.3482	1.0710	1.2278	1.1485
CNN	0.9968	0.9965	0.9969	0.8148	0.8249	0.8015	0.6387	0.6290	0.6680
CNN-LSTM	0.9982	0.9982	0.9981	0.6422	0.6270	0.6292	0.5043	0.4726	0.5048

Table 3. Comparison analysis between various models for simulated monthly average atmospheric temperature.

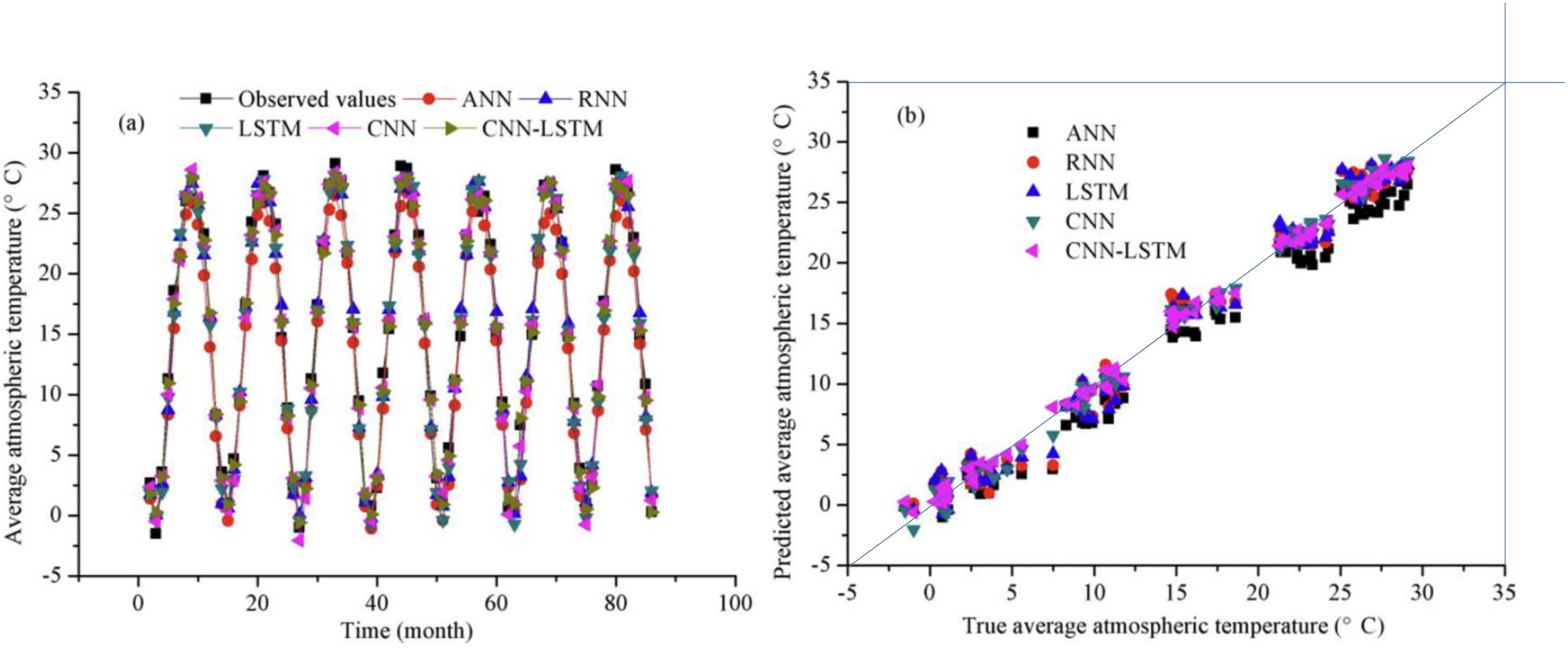


Figure 6. Comparison of observed and predicted values of monthly average atmospheric temperature in Jinan from 2015 to 2022. (**a**) Plot of the prediction results, (**b**) Scatterplot of the prediction results.

Models	R			RMSE (mm)			MAE (mm)		
	Training	Verification	Predicting	Training	Verification	Predicting	Training	Verification	Predicting
ANN	0.6853	0.6823	0.7092	58.7736	56.3767	67.4976	39.8891	34.4801	42.5787
RNN	0.7412	0.7452	0.7590	53.0472	50.2847	62.1261	33.1355	29.3883	37.7565
LSTM	0.7685	0.7246	0.7672	50.6978	52.4807	60.5523	32.5079	31.4153	36.0748
CNN	0.9691	0.9629	0.9857	20.2432	22.5777	16.8436	13.8417	13.8229	11.8683
CNN-LSTM	0.9952	0.9930	0.9962	7.8252	8.8947	8.1762	6.0644	6.7083	6.7051

Table 6. Comparison analysis between various models for simulated monthly precipitation.

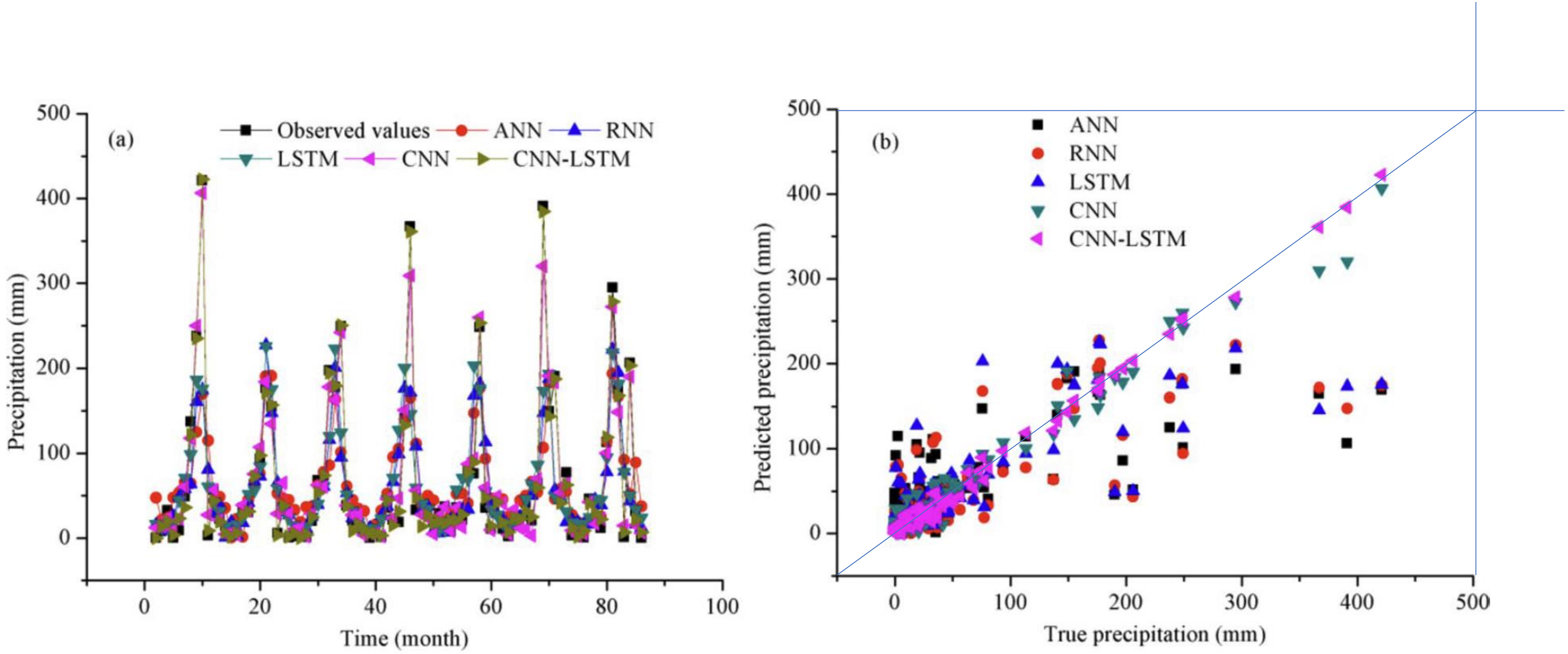


Figure 9. Comparison of observed and predicted values of monthly precipitation in Jinan from 2015 to 2022. (**a**) Plot of the prediction results, (**b**) Scatterplot of the prediction results.

Models	ANN	RNN	LSTM	CNN	CNN-LSTM
Average elapsed time	17	18	19	40	89

Table 9. Average time for each model run.

Python代码演示

IMDB Document Classification

We now implement models for sentiment classification (Section???~~) on the `IMDB` dataset. As mentioned above code block~~8, we are using a preprocessed version of the `IMDB` dataset found in the `keras` package. As `keras` uses `tensorflow`, a different tensor and deep learning library, we have converted the data to be suitable for `torch`. The code used to convert from `keras` is available in the module `ISLP.torch._make_imdb`. It requires some of the `keras` packages to run. These data use a dictionary of size 10,000.

We have stored three different representations of the review data for this lab:

- `load_tensor()`, a sparse tensor version usable by `torch`;
- `load_sparse()`, a sparse matrix version usable by `sklearn`, since we will compare with a lasso fit;
- `load_sequential()`, a padded version of the original sequence representation, limited to the last 500 words of each review.

+ 31 cells hidden

Recurrent Neural Networks

In this lab we fit the models illustrated in Section~???.

+ 56 cells hidden

Click to add a cell.