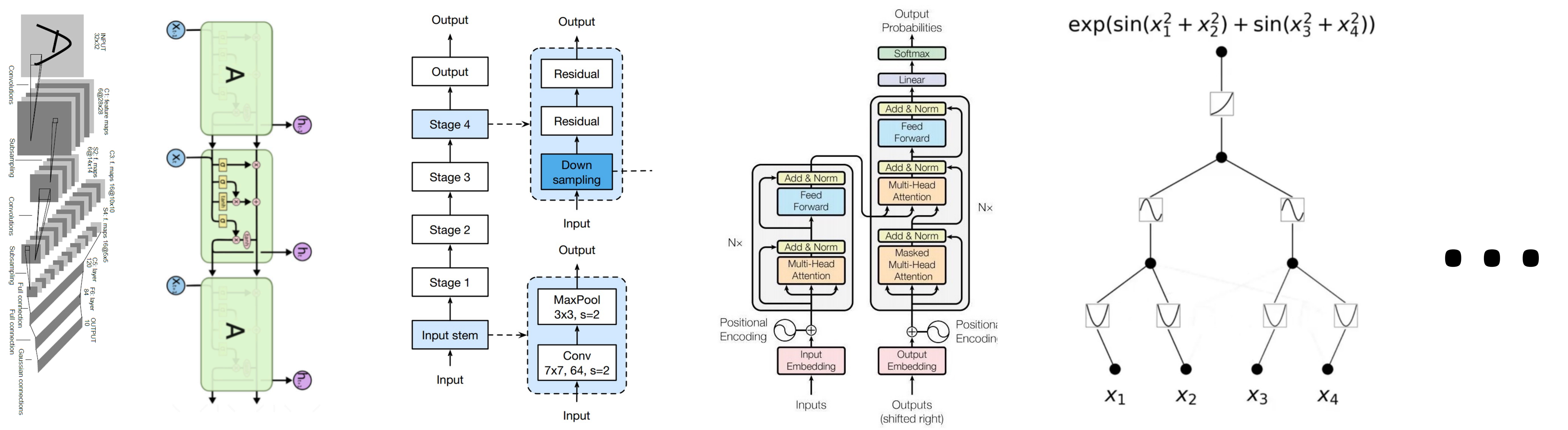
DeepMovie: Plug-and-Play DNN Training Pipeline for Movie Recommendation

Code Available

Jiazi Bu[*1,2], Zhiyuan Zhang[*1,2], Xizhuo Zhang[*1,2]
[1]Shanghai Jiaotong University [2]School of Electronic Information and Electrical Engineering [*]Equally Contribution

DeepMovie can integrate with many of the most advanced deep learning modules in a plug-and-play manner

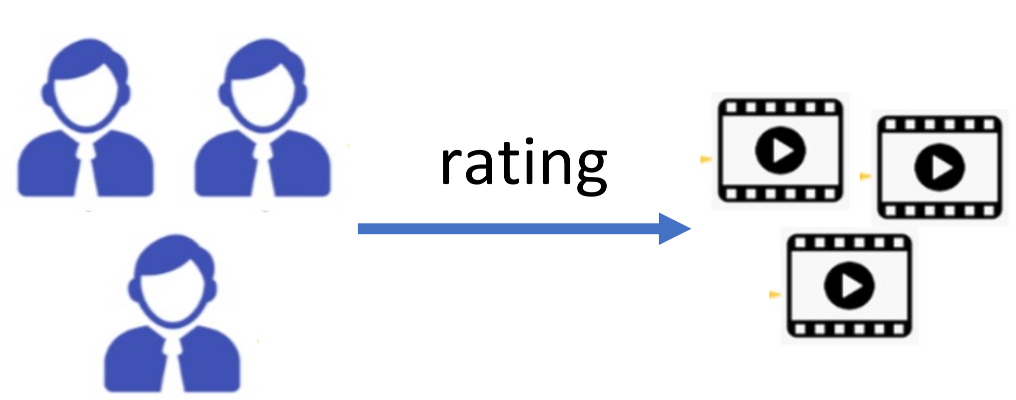
DeepMovie 🔥



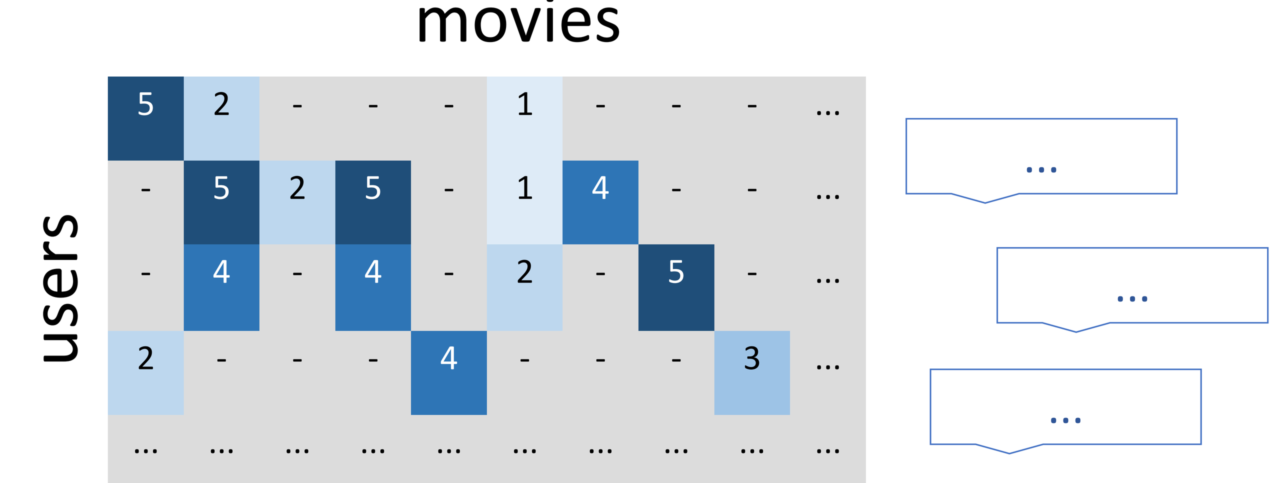
CNN LSTM ResNet Transformer KAN and more

Overview of the DeepMovie pipeline

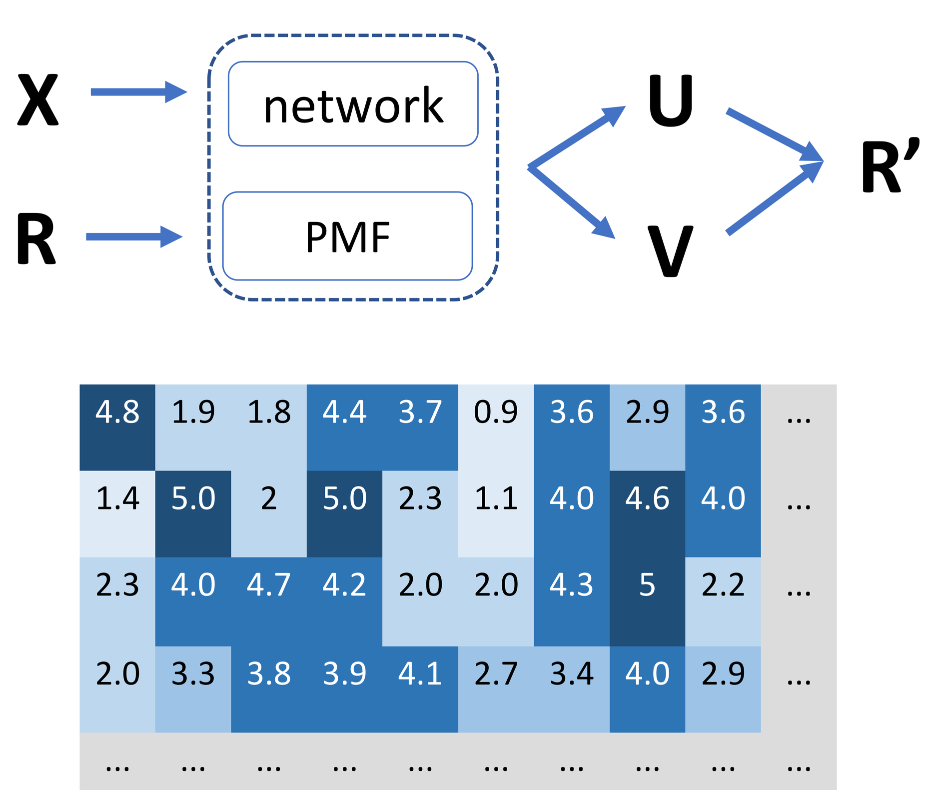
DEEPMOVIE



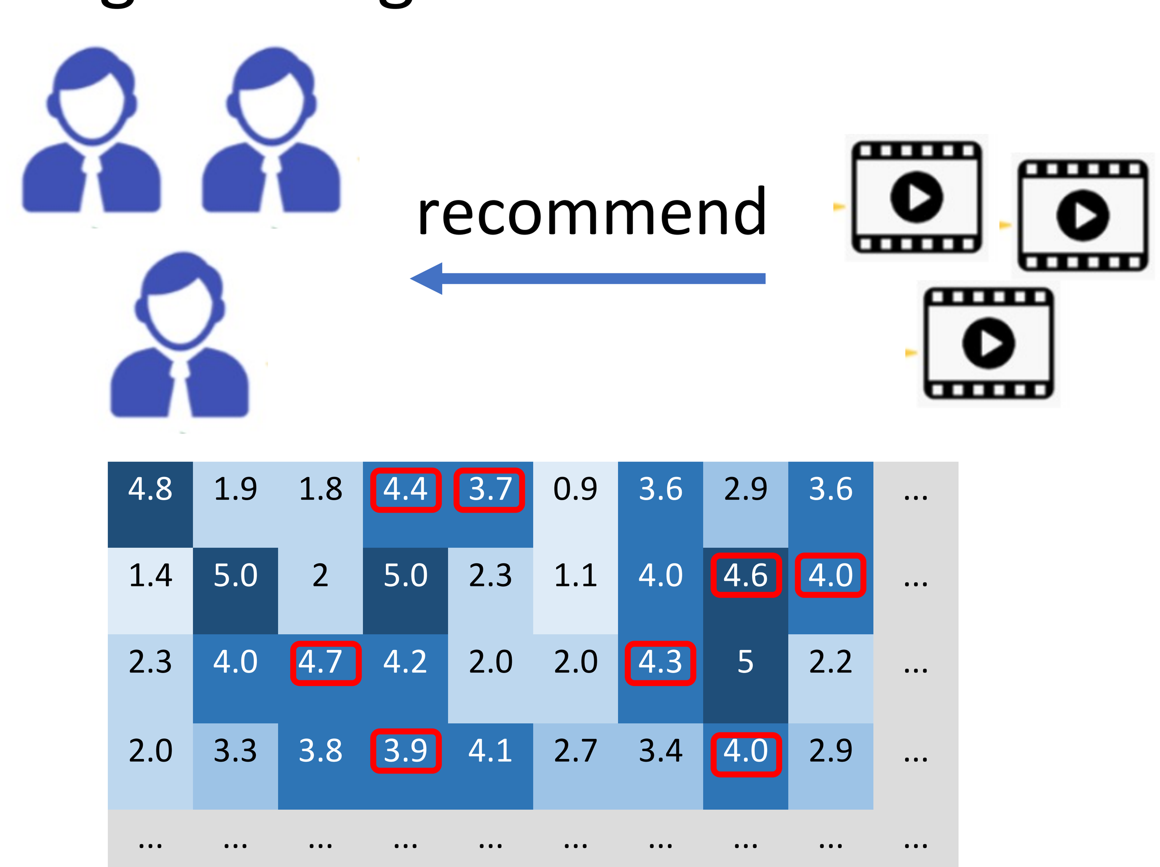
1.user rates some movies (about 20)



2.get ratings R and reviews X



3.get latent matrix with our model and reconstruct ratings



4.recommend movies for each user with reconstructed ratings

Experimental results on MovieLens dataset

- Performance metrics (in terms of RMSE loss) of DeepMovie based on different base modules on MovieLens-1M and 10M benchmark

Base Module	MovieLens-1M	MovieLens-10M
PMF (baseline)	0.8971	0.8311
CNN (+ MLP)	0.8733	0.7970
LSTM	0.8675	0.7959
ResNet	0.8658	0.7931
Transformer	**0.8601**	**0.7883**
CNN + KAN	0.8725	0.7941

- Single epoch training time of different base modules (single A10 GPU)

CNN	LSTM	ResNet	Transformer	KAN
2.7720	3.8126	4.2604	10.9185	3.1637

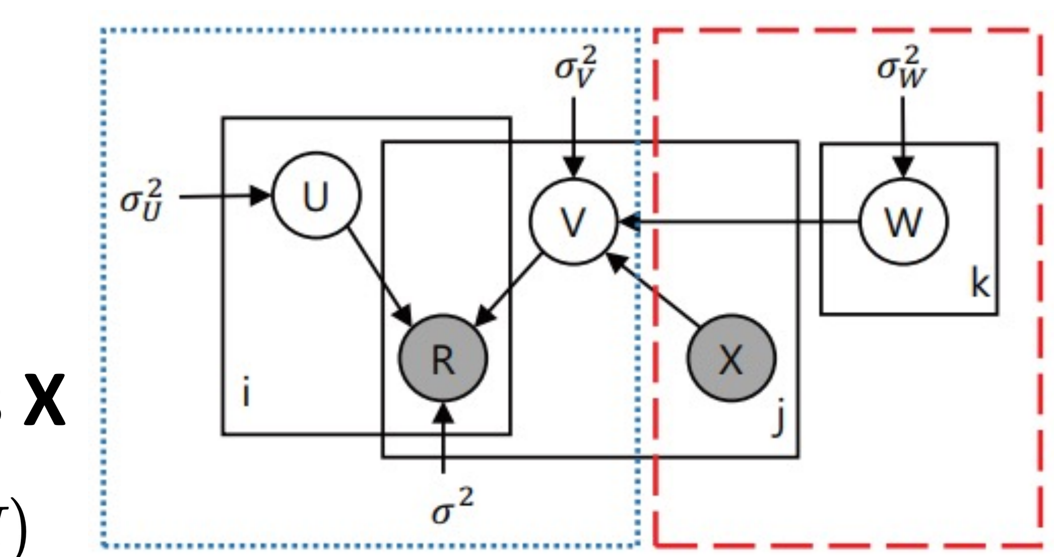
Method: Integrates neural network into PMF

- probabilistic matrix factorization (PMF)
- find latent models of users and items on a shared latent space

$$R \approx U^T V$$

large but sparse → small and dense

R: user-item rating matrix, NxM
U: user latent matrix, KxN
V: item latent matrix, KxM



- suppose Gaussian observation noise

$$p(R \mid U, V, \sigma^2) = \prod_i^N \prod_j^M N(r_{ij} \mid u_i^T v_j, \sigma^2)^{I_{ij}}$$

- use network to extract **V** from reviews **X**

$$p(V \mid W, X, \sigma_V^2) = \prod_j^M N(v_j \mid network(W, X_j), \sigma_V^2 I)$$

- Optimize through maximum a posteriori(MAP)

$$\max_{U,V,W} p(U, V, W \mid R, X, \sigma^2, \sigma_U^2, \sigma_V^2, \sigma_W^2)$$

$$\min\left(\|I \otimes (R - U^T V)\|_{Fro} + \lambda_V \|V - network(W, X)\|_{Fro} + \lambda_U \|U\|_{Fro} + \lambda_V \|V\|_{Fro}\right)$$

- Update

1. update **U**: $u_i \leftarrow (V I_i V^T + \lambda_U I_K)^{-1} V R_i$
2. update **V**: $v_j \leftarrow (U I_j U^T + \lambda_V I_K)^{-1} (U R_j + \lambda_V network(W, X_j))$
3. train the network to fit **V**

repeat until converge