Session 3:
VisRec: A Hands-on Tutorial on Deep Learning for Visual Recommender Systems

Denis Parra, Antonio Ossa-Guerra, Manuel Cartagena, **Patricio Cerda-Mardini***, Felipe del Río
Pontificia Universidad Católica de Chile

* mindsdb

26th ACM Conference on Intelligent User Interfaces IUI 2021

Visually-Aware Fashion Recommendation and Design with Generative Image Models

Wang-Cheng Kang
UC San Diego
wckang@eng.ucsd.edu

Chen Fang
Adobe Research
cfang@adobe.com

Zhaowen Wang
Adobe Research
zhawang@adobe.com

Julian McAuley
UC San Diego
jmcauley@eng.ucsd.edu

a.k.a. DVBPR:
Deep Visually-aware Bayesian Personalized Ranking

About the paper

- Accepted at the International Conference on Data Mining (ICDM), 2017
- Authors: Adobe Research & Prof. McAuley’s Lab @ UCSD
- Proposes fashion 1) recommendation and 2) design (through GANs)
- We focus on 1)

Methodology

Source: Kang et al. 2017

Context

- Fashion domain is complex: long tails, cold starts, evolving dynamics

- Content-aware recommender systems are well-suited to it

Source: Kang et al. 2017

DVBPR Key Insights

- Opt for “domain-aware” visual embeddings instead of “off-the-shelf” as in VBPR
- Joint training of visual embeddings and recommender system
- Generate new items consistent with each user’s preference

Approach

- BPR framework: optimize rank of purchased vs non-purchased items

- Siamese trainable CNNs contrast positive and negative pairs
    - Images are retrieved and rescaled in the DataLoader

- Original datasets: Amazon fashion + Tradesy

- In this tutorial: Wikimedia Commons dataset

Model

- Users $u \in \mathcal{U}$
- Items $i \in \mathcal{I}$
- Positive items $\mathcal{I}_u^+$
- Item image X_i

VBPR:

$$x_{u,i} = \beta_i + \gamma_u^T \gamma_i + \theta_u^T (E \cdot f_i)$$

DVBPR:

$$x_{u,i} = \beta_i + \gamma_u^T \gamma_i + \theta_u^T \phi(X_i)$$

Model

VBPR:

Item bias | Latent CF factors | Visual factors

$$x_{u,i} = \beta_i + \gamma_u^T \gamma_i + \theta_u^T (E \cdot f_i)$$

Dimensionality reduction | Pre-trained embedding

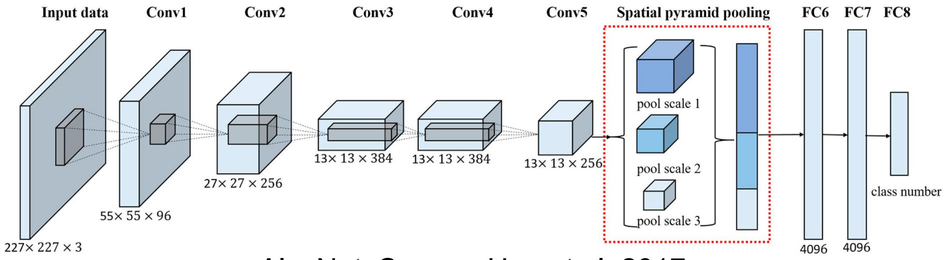
DVBPR:

$$x_{u,i} = \beta_i + \gamma_u^T \gamma_i + \theta_u^T \phi(X_i)$$

We keep this! | Trainable CNN

Convolutional Neural Networks

- We use AlexNet with K=100



AlexNet. Source: Han et al. 2017

- Paper uses CNN-F with K=50

conv1	conv2	conv3	conv4	conv5	full6	full7	full8
64x11x11	256x5x5	256x3x3	256x3x3	256x3x3	4096	4096	K
st. 4, pad 0	st. 1, pad 2	st. 1, pad 1	st. 1, pad 1	st. 1, pad 1	drop-	drop-	-
x2 pool	x2 pool	-	-	x2 pool	out	out	-

CNN-F. Source: Kang et al. 2017

BPR Optimization

$$u \in \mathcal{U} \qquad i \in \mathcal{I}_u^+ \qquad j \in \mathcal{I} \setminus \mathcal{I}_u^+ \qquad (u, i, j) \in \mathcal{D}$$

$$\mathcal{D} = \{(u, i, j) | u \in \mathcal{U} \wedge i \in \mathcal{I}_u^+ \wedge j \in \mathcal{I} \setminus \mathcal{I}_u^+\}$$

$$\max \sum_{(u,i,j) \in \mathcal{D}} \ln \sigma(x_{u,i,j}) - \lambda_\Theta \|\Theta\|^2$$

$$x_{u,i,j} = x_{u,i} - x_{u,j}$$

Retrieval / Recommendation

$$\delta(u,c) = \underset{i \in X_c}{\operatorname{argmax}}\ x_{u,i} = \underset{i \in X_c}{\operatorname{argmax}}\ \beta_i + \gamma_u^T \gamma_i + \theta_u^T \phi(X_i)$$

Observations

- Model converges after 5 epochs (~12 hours on an 8-core CPU + GTX 1080 Ti)

- In our experience, latent CF factors are crucial for the model to learn

Datasets

Dataset	# Users	# Items	# Interactions	# Categories
Amazon Fashion	64583	234892	513367	6
Amazon Women	97678	347591	827678	53
Amazon Men	34244	110636	254870	50
Tradesy.com	33864	326393	655409	N / A
Wikimedia	1078	32959	96991	N / A

- Wikimedia interactions are related to image quality

Implementation details - Model Class

```python
class DVBPR(nn.Module):
    def __init__(self, n_users, n_items, K=2048, use_cnnf=False):
        super().__init__()
        self.cache = None

        # CNN for learned image features
        if use_cnnf:
            self.cnn = CNNF(hidden_dim=K)  # CNN-F is a smaller CNN
        else:
            alexnet = models.alexnet(pretrained=False)
            final_len = alexnet.classifier[-1].weight.shape[1]
            alexnet.classifier[-1] = nn.Linear(final_len, K)
            self.cnn = alexnet

        # Visual latent preference (theta)
        self.theta_users = nn.Embedding(n_users, K)

        # Latent factors (gamma)
        self.gamma_users = nn.Embedding(n_users, 100)
        self.gamma_items = nn.Embedding(n_items, 100)

        # Random weight initialization
        self.reset_parameters()
```

Implementation details - Forward Pass

```python
def forward(self, ui, pimg, nimg, pi, ni):
    # User
    ui_visual_factors = self.theta_users(ui)  # Visual factors of user u
    ui_latent_factors = self.gamma_users(ui)  # Latent factors of user u

    # Items
    pi_features = self.cnn(pimg)  # Pos. item visual features
    ni_features = self.cnn(nimg)  # Neg. item visual features

    pi_latent_factors = self.gamma_items(pi)  # Pos. item visual factors
    ni_latent_factors = self.gamma_items(ni)  # Neg. item visual factors

    x_ui = (ui_visual_factors * pi_features).sum(1) + (pi_latent_factors * ui_latent_factors).sum(1)
    x_uj = (ui_visual_factors * ni_features).sum(1) + (ni_latent_factors * ui_latent_factors).sum(1)

    return x_ui, x_uj
```

Implementation details - Optimization

```
# Forward pass
with torch.set_grad_enabled(phase == "train"):
    pos, neg = self.model(profile, pimg, nimg, pi, ni)
    output = pos-neg
    loss = self.criterion(output, target)
    loss += (1.0 * torch.norm(self.model.theta_users.weight))

    # Backward pass
    if phase == "train":
        loss.backward()
        self.optimizer.step()
```

Implementation details - Recommendation

```python
def recommend_all(self, user, cache, grad_enabled=False):
    with torch.set_grad_enabled(grad_enabled):
        # User
        u_visual_factors = self.theta_users(user)  # Visual factors of user u
        ui_latent_factors = self.gamma_users(user)

        # Items
        i_latent_factors = self.gamma_items.weight  # Items visual factors

        x_ui = ((i_latent_factors * ui_latent_factors).sum(dim=1).squeeze() + \
                (u_visual_factors * cache).sum(dim=2).squeeze())

        return x_ui
```

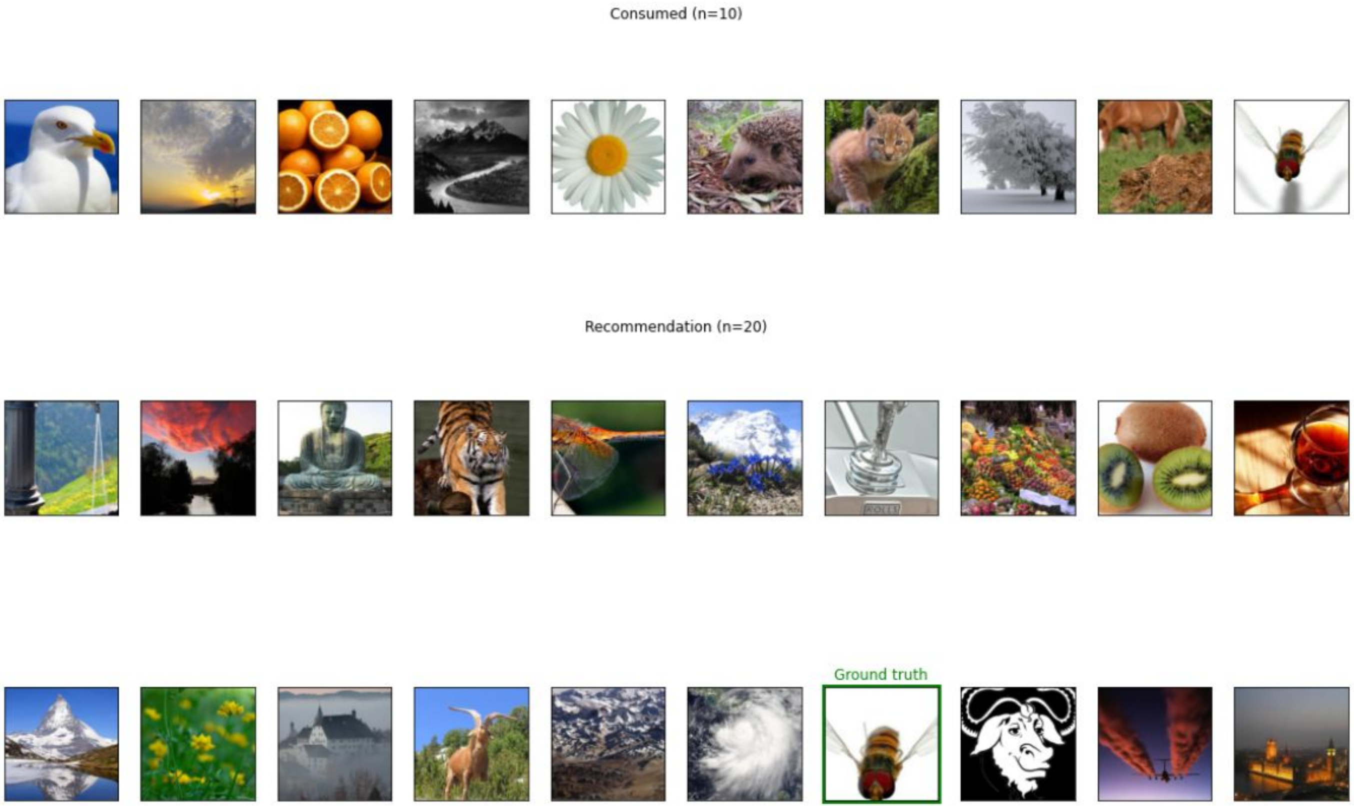
Main Results

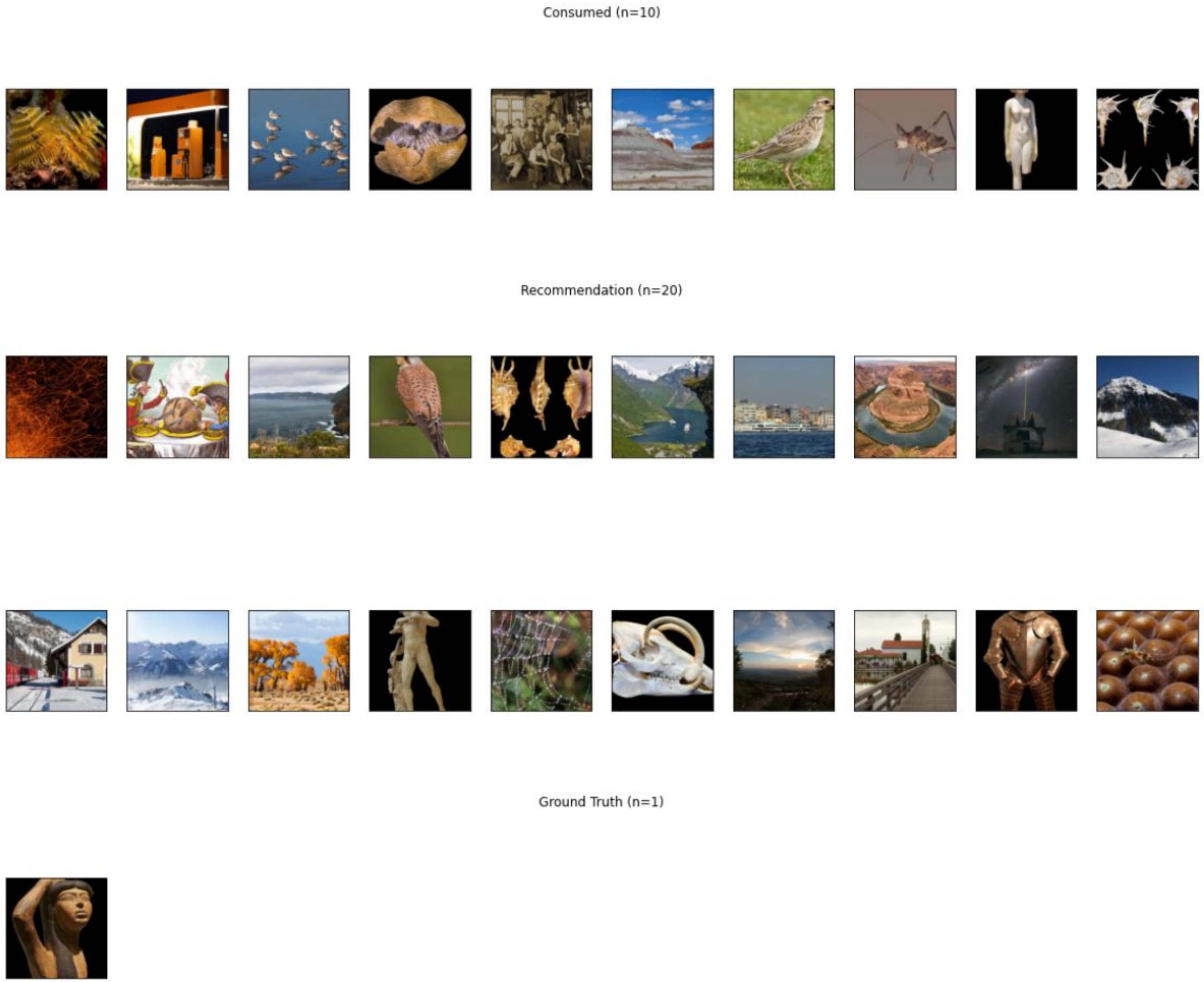
AUC	RR	R@20	P@20	nDCG@20	R@100	P@100	nDCG@100
0.83169	0.04507	0.12152	0.00608	0.05814	0.25696	0.00257	0.08245

- Better AUC than in Tradesy and Amazon

- Best Wikimedia performer out of 4 architectures presented in this tutorial

Example recommendations




Consumed (n=10)
Recommendation (n=20)
Ground Truth (n=1)

Conclusions

- Wikimedia dataset is different from Amazon and Tradesy
    - Image quality is primary concern
    - Content < Collaborative Filtering

- This might explain why latent non-visual factors are needed

- DVBPR approach is simple yet effective for visual recommendation in challenging domains

- Newer CNN architectures might be interesting to explore
    - EfficientNet
    - Lambda Networks

References

[1] Kang, W., Fang, C., Wang, Z., & McAuley, J. (2017). Visually-Aware Fashion Recommendation and Design with Generative Image Models. *2017 IEEE International Conference on Data Mining (ICDM)*, 207-216.

[2] Krizhevsky, A., Sutskever, I., & Hinton, G.E. (2012). ImageNet classification with deep convolutional neural networks. *Communications of the ACM, 60*, 84 - 90.

[3] Chatfield, K., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Return of the Devil in the Details: Delving Deep into Convolutional Nets. *ArXiv, abs/1405.3531*.

Hands-on!

Now, let’s (briefly) check out the notebook

Session 3:
VisRec: A Hands-on Tutorial on Deep Learning for Visual Recommender Systems

Denis Parra, Antonio Ossa-Guerra, Manuel Cartagena, **Patricio Cerda-Mardini***, Felipe del Río
Pontificia Universidad Católica de Chile

* mindsdb

26th ACM Conference on Intelligent User Interfaces IUI 2021