An Application to Generate Style Guided Compatible Outfit

*Debopriyo Banerjee[1,2], *Harsh Maheshwari[1], *Lucky Dhakad[1], Arnab Bhattacharya[1], Niloy Ganguly[2], Muthusamy Chelliah[1], Suyash Agarwal[1]*
deb.ban89@gmail.com
[1]Flipkart Internet Pvt. Ltd., [2]Indian Institute of Technology Kharagpur
India *equal contributions

ABSTRACT

Fashion recommendation has witnessed a phenomenal growth of research, particularly in the domains of shop-the-look, context-aware outfit creation, personalizing outfit creation etc. Majority of the work in this area focuses on better understanding of the notion of complimentary relationship between lifestyle items. Quite recently, some works have realised that *style* plays a vital role in fashion, especially in the understanding of compatibility learning and outfit creation. In this paper, we would like to present the end-to-end design of a methodology in which we aim to generate outfits guided by styles or themes using a novel style encoder network. We present an extensive analysis of different aspects of our method through various experiments. We also provide a demonstration api to showcase the ability of our work in generating outfits based on an anchor item and styles.

CCS CONCEPTS

• **Computing methodologies** → **Learning latent representations**; **Neural networks**.

KEYWORDS

complete the look, neural networks, outfit compatibility, style

ACM Reference Format:
*Debopriyo Banerjee[1,2], *Harsh Maheshwari[1], *Lucky Dhakad[1], Arnab Bhattacharya[1], Niloy Ganguly[2], Muthusamy Chelliah[1], Suyash Agarwal[1]. 2022. An Application to Generate Style Guided Compatible Outfit. In *5th Joint International Conference on Data Science & Management of Data (9th ACM IKDD CODS and 27th COMAD) (CODS-COMAD 2022), January 8–10, 2022, Bangalore, India.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3493700.3493737

1 INTRODUCTION

Outfit recommendation is a relatively well studied area in which researchers aim to recommend outfits based on the notion of *compatibility* between lifestyle items, see [11, 17–19] for more details. A substantial volume of work has also been done on the specific area of personalised recommendations [13, 20]. However, none of them specifically take style into account while learning compatibility within outfits. We realise that style is an essential component in modelling outfit compatibility as an outfit may look compatible under one style construct but not in another.

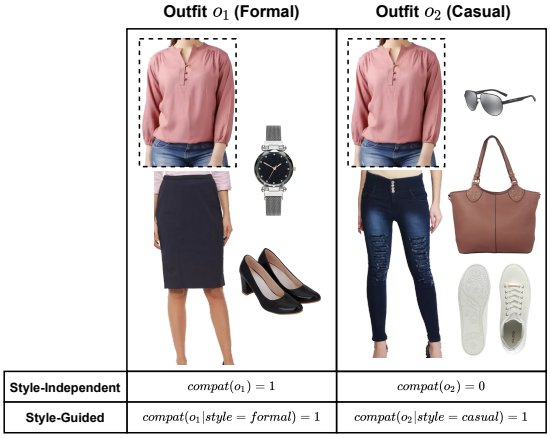

Figure 1: Illustration of the effectiveness of style-guided outfit generation over a style-independent variant. If *formal* is the dominant style, the latter will only accept outfits from *formal* style while rejecting from others. Style-guided methods, however, will accept outfits from multiple styles.

An example of style guided outfit creation is provided in Figure 1. The same top item (highlighted by a dotted rectangle) is used to create two outfits under two different styles, namely *Formal* and *Casual*. This is useful in the situation where we consider that a user likes the top item but is doubtful about making the final purchase. A style-guided algorithm will have two advantages: (a) it will reject an outfit which may be otherwise compatible but not in accordance with the desired style, and (b) it will pick compatible outfits from different styles, hence expanding the choice to the user. A style-independent algorithm, on the other hand, gets biased towards the dominant style (assuming *formal* style is dominant).

Style guided recommender system requires special attention from the research community as most of the work are unsupervised in nature. Some example research in this area are listed below: (a). Kuhn et al. [7] refer an outfit as a style fit and do not explicitly use style information for modeling compatibility; (b). Jeon et al. [5] extract fashion attributes from full-body outfit images for classifying outfit style; (c). Li et al. [10] models outfit level style from item descriptions; (d). Singhal et al. [16] models context and type jointly using Graph Neural Network (GNN), and style between item pairs are modeled using autoencoder without any explicit style information. Each of these works lack in one way or the other the ability to generate style guided outfits.

Lai et al. proposed the Theme Matters paper [8] (archived work) which comes closest to our work. It projects a supervised approach

*First three authors contributed equally to this research.

CODS-COMAD 2022, January 8–10, 2022, Bangalore, India

ACM ISBN 978-1-4503-8582-4/22/01.
https://doi.org/10.1145/3493700.3493737

that applies theme-aware attention to item pairs with fine-grained category tags (e.g., long-skirt, mini-skirt, t-shirt, etc.). There are two specific drawbacks, the first of which is that such fine-grained category information is not always available and can be ambiguous if done manually. Secondly, the size of the model increases exponentially with the number of fine-grained categories.

We propose a **Style-Attention-based Compatible Outfit Generation** (SATCOGen) framework that uses high-level categories (e.g., topwear, bottomwear, footwear, accessory, etc.) and outfit-level style information. It consists of a Style-Compatibility-Attention Network (SCA Net) [12] and a novel style encoder network called Variational Style Encoder Network (VSEN) which encodes the style of an outfit into a latent space. This encoding is used to provide style-specific subspace attention, along with category information during the computation of embedding. Multiple loss functions ensure style encoding, general as well as style-specific compatibility. For the generation task, given an anchor item, beam search is used to generate style-specific outfit. We have provided a demonstration api to showcase the kind of outfits that are generated for an anchor item given various styles.

2 METHODOLOGY

The fundamental philosophy guiding the work in this paper is that in practical circumstances compatibility between lifestyle items present within an outfit is contingent on the style to which the outfit belongs. In a nutshell, we make use of Style-Compatibility-Attention Network (SCA Net), a compatibility learning framework that makes use of features extracted from the image of an item based on category information as previously done by [12] and then add style component to it. The methodology of SATCOGen is explained in greater detail below.

A smooth latent vector representation for outfit style is learnt using Variational inference in a novel style encoder named *Variational Style Encoder Network* (VSEN). There are two main trends in denoting an outfit, as an ordered sequence of items [4, 15] or as *set* [1, 3]. We choose the latter representation, which brings in two important properties, namely permutation invariance and allowance for varying length. This assumption enables us to select the *set transformer* approach proposed in [9] for our style encoder job. Keeping in mind that our work is not restricted only to compatibility learning and also involves outfit generation, we ensure that every outfit style is represented by the first two moments of a Gaussian distribution which is proximal to the unit Gaussian $\mathcal{N}(0, \mathbb{1})$, a mechanism we borrowed from Variational inference [2]. This further ensures smooth representation of the latent style space. The advantage of this step will be clear during the outfit generation stage.

A vector, sampled from the Gaussian distribution representing the style of the outfit is used to classify the style of the outfit. This ensures that VSEN is able to capture specific information about the style of the outfit as well. Thus, given the styles and their corresponding style vectors, this module solves a multi-class classification problem using an MLP with n layers.

We modify the subspace attention network proposed in [12] to learn compatibility between items in an outfit. In the previous network, the image of an item within an outfit is passed through a ResNet18, and the embedding vector output is multiplied by learnt masks that help to learn the subspaces. The item and target categories are then consumed to estimate the subspace attention weights which subsequently leads to a weighted average of the masked embeddings to be denoted as the final embedding of the item in the tuple `<item, item category, target category>`. We tweak this and estimate the subspace attention weights by providing the outfit specific sample style vector from VSEN as an additional input. This helps to learn compatibility conditional on the style of the outfit.

Figure 2: Beam search: Given a top-wear chosen by a user, and a template, the algorithm would go about generating outfits by sequentially adding items from each category in the template.

There are four loss functions used in our method for learning style specific compatibility. We have the KL divergence loss from the VSEN network and the classification loss from the downstream job. We also have the compatibility loss from the SCA Net which is based on the popular triplet loss. And finally, we introduce one more loss function to account for penalisation when the wrong style is specified for an outfit. The overall loss for our method is given as the weighted sum of these four individual losses.

2.1 Outfit generation

A globally optimal outfit generation task is a non-trivial task since it is infeasible to look into all possible combinations. An approximate solution is provided in this case. First, embeddings are created for different target categories for an item and an associated style. Note that we know from the previous section that embedding computation requires us to provide a style vector for every item. If there is a reference outfit from the same style which we want to emulate, it is trivial to generate a style vector from VSEN using that outfit. However, in the absence of a reference outfit, there is no specific distribution to sample from. For this we pool the mean and variance of all outfits belonging to that specific style, and use a Gaussian with pooled moments. This distribution can be assumed to be representative of the style in question and enables us to generate a style vector from it.

Once we have estimated embeddings for each item, one can generate outfits based on the well known *beam search method* [21], as is shown in Figure 2.

3 DATASET AND METRICS

We created a female outfit dataset with style annotations.

Table 1: Distribution of compatible outfits across different styles.

Style	Work	Casual	Party	Relax	Travel	Athleisure	Sporty
Train	841	13062	1215	473	2128	1160	534
Val	108	1679	156	61	272	149	68
Test	251	3917	362	140	631	348	160

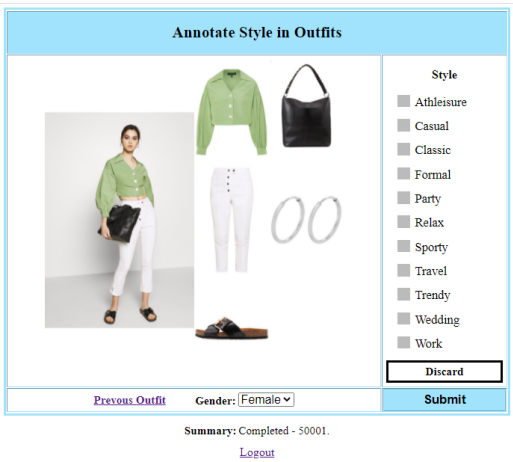

Figure 3: Annotation interface for curating the style of Zalando data. A complete outfit representing a *look* from the website is provided along with individual items from the outfit. The annotator has to assign one or more appropriate style tags to the outfit, along with selecting the suitable gender for it. One can also discard the item if there is any form of discrepancy.

Zalando Dataset (Zal): This dataset consists of items and outfits from the Zalando website[1]. In the website, *outfit looks* are displayed with the option of shopping items from them. We scraped the looks and the corresponding set of items present in each of the looks. Since style tags are unavailable for majority of the looks, we recruit human annotators to perform the task of annotating style tags in outfits. We present the annotation interface in figure 3, where we displayed an outfit look image along with the set individual items in the look on the left and center and a list of style options on the right. The annotator can assign one or more appropriate style tags to each outfit. We also provided a dropdown list to select the gender for which the outfit is primarily suitable. In case there is a mismatch in gender of the outfit look and the associated items or any other discrepancies, the annotator has the option to select *Discard*. It is ensured that each outfit is shown only once to each annotator.

This dataset consists of items and outfits from the Zalando website (https://www.zalando.co.uk/). In the website, *outfit looks* are displayed with the option of shopping items from them. We merged semantically similar fine-grained item categories, which resulted in nine higher level categories. We scraped the looks and the corresponding set of items present in each of the looks. Since style tags are unavailable for majority of the looks, we recruit human annotators to perform the task of annotating style tags in outfits. We present the annotation interface in figure 3, where we displayed an outfit look image along with the set individual items in the look on the left and center and a list of style options on the right. The annotator can assign one or more appropriate style tags to each outfit. We also provided a dropdown list to select the gender for which the outfit is primarily suitable. In case there is a mismatch in gender of the outfit look and the associated items or any other discrepancies, the annotator has the option to select discard. Finally, there is an option to edit the immediately previous outfit and the number of completed annotations are shown in summary. It is ensured that each outfit is shown only once to each annotator.

Even though an outfit can have multiple style tags, for this work we ensure that each outfit has a single style. In the situation that an outfit has multiple style tags from the two annotations, we choose the one with the higher vote. Ties are broken by randomly selecting a style. After the annotation task, it was found that some of the styles were heavily under-represented. To mitigate this issue, we merged certain styles that are semantically similar and discarded some as well. For example, *party* and *wedding* were merged into *party* while all outfits from *classic* and *trendy* were discarded. At the end we had ~28K outfits.

Metrics: Two well known metrics are used to evaluate the performance of an outfit compatibility prediction model [4, 14].
Fill-in-the-blank Accuracy (FITB Acc.): Given a set of items of an outfit with one missing item as query, the task is to predict the correct missing item from a list of four option items (where one is correct and three are incorrect) based on compatibility of each option item with the query set.
Compatibility AUROC (Compat. AUC): Given a set of positive and negative outfit samples, this metric helps in measuring the quality of predictions for compatible and incompatible outfits.

We constructed separate test sets for FITB and compatibility tasks similar to [18] by creating soft negative (SN) and hard negative (HN) samples corresponding to each positive outfit sample. In case of SN, we sample random items from the matching higher level categories (e.g., topwear, bottomwear, footwear, etc.), whereas for HN, we do the sampling from matching fine-grained categories[2] (e.g., t-shirts, heels, shoes, etc.). It is to be noted that *HN* samples are relatively harder to differentiate from positives than SN samples. This gradation helps in evaluating the performance of SATCOGen at various difficulty levels. We consider five FITB and Compatibility test datasets and report the mean performance.

4 IMPLEMENTATION DETAILS

The two modules of SATCOGen, namely VSEN and SCA Net make use of RestNet18 as the backbone CNN architecture. We freeze all the layers of ResNet18 except the last convolutional block, which is connected to a new fully connected layer that outputs a 64 dimensional embedding vector similar to the state-of-the-art compatibility learning methods [12, 18].

VSEN aggregates the CNN features of all the items in an outfit using the SAB Set Transformer [9] with a hidden dimension of 32 and two heads for mean and variance. The output of style vector

[1]https://www.zalando.co.uk/

[2]We make use of fine-grained category information only during evaluation or testing phase.

Style Pre-conditioning	Bottomwear 1	Bottomwear 2
Athleisure	0.2621	0.2525
Casual	0.2550	0.2697

Figure 4: Demonstration of how SATCOGen is able to choose diverse style relevant bottomwears for a given parent topwear.

of VSEN has a dimension of 64 which is followed by two fully connected MLP layers for the style classification task. Here we consider batch size as 128 and employ the Adam optimizer [6] with an initial learning rate of 5×10^{-6}. When combining the KL-Divergence loss with the cross entropy loss of classification, we consider 0.05 as the weight coefficient corresponding to KL-Divergence loss. After training the VSEN module, we freeze all of its parameters and use it in evaluation mode to train the rest of the SATCOGen framework. For learning the parameters of the SCA Net (five subspaces), we use the Adam optimizer with batch size of 32 triplets, and initial learning rate of 1×10^{-6}. The Attention Network of SCA Net transforms the concatenated one-hot-encoded category vectors and the extracted style vectors from VSEN into 32 dimensions, respectively (using a single fully connected layer), which are then forwarded to two fully connected layers (after concatenation). Finally, the output is the five subspace attention weights.

For outfit generation, we have optimized the inference code using native spark implementation to get item embedding given style and category information and to run beam search at scale on a large scale of volume of anchor and candidate items, resulting in 20X run time reduction compared to single box implementation. We compared the run time on 60K anchor items with 30K average child candidates per category, 5 average categories in beam search template and beam width of 3.

Table 2: Comparison of compatibility learning for different methods on the Zalando dataset. We compute FITB and Compatibility ROC AUC with both hard and soft negatives.

Method	Type	FITB Acc.	Compat. AUC
TM	SN	47.79 ± 0.07	76.73 ± 0.06
	HN	43.78 ± 0.25	75.97 ± 0.08
SATCOGen	SN	59.10 ± 0.34	88.58 ± 0.08
	HN	55.90 ± 0.31	86.96 ± 0.06

5 RESULTS

We show the efficacy of SATCOGen in generating superior outfits for online shopping portals by comparing its performance against Theme Matters (TM) [8] based on FITB Acc. and Compat. AUC scores. Table 2 presents the FITB Acc. and Compat AUC results on SN and HN samples of Zal dataset for TM and SATCOGen (our proposed model). SATCOGen outperforms the TM results, mainly because of the requirement of fine-grained category information to have improved performance. Fig. 4 shows an example of the quality of parent-child category combinations with style pre-conditioning.

6 DEMONSTRATION INTERFACE

We have created a demonstration api to showcase the outfits generated by using SATCOGen framework given an anchor item, a style and candidate items from other categories. The outfit templates have been identified after discussion with fashion experts. An example of such template would be, *('dress', 'heels', 'bag', 'jewellery')*. In the api, we provide users to select a category and subsequently an anchor item from the category for which the user wants to view outfits under different legitimate styles. The top-5 outfits per style are displayed for the selected anchor item. In figure-5, we are showcasing the top-3 outfits for an anchor t-shirt given style *sporty* and template *'t-shirt', 'leggings', 'trainer-shoes', 'bra'*. The demo-api along with details and screenshots can be found here [3].

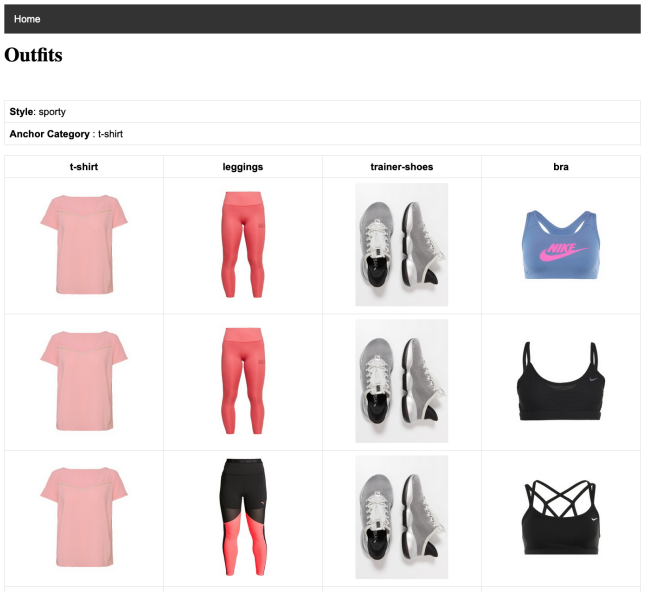

Figure 5: Screenshot of the Web Interface used for Demonstration.

7 CONCLUSION

In this paper, we presented SATCOGen - a novel outfit generation framework based on styles and evaluated its performance on Zal - a newly introduced outfit dataset with style tags associated with each outfit. In general, the outfit generation process using beam search algorithm is time consuming and not scalable for datasets having thousands of items for each category. We ensured scalability, by optimizing the code and making it suitable for execution on Hadoop Clusters, which reduced the execution time drastically. Finally, we presented a web-interface that demonstrates outfit generation starting from a chosen anchor item, a pre-defined style, and a template. In the future, we are planning to extend the outfit dataset with ethnic outfits (specific to Indian context) and productionize the SATCOGen framework to provide an enhanced shopping experience to the customers.

[3] https://github.com/Lucky-Dhakad/SATCOGen-Demo-api

REFERENCES

[1] Elaine M. Bettaney, Stephen R. Hardwick, Odysseas Zisimopoulos, and Benjamin Paul Chamberlain. 2019. Fashion Outfit Generation for E-commerce. arXiv:1904.00741

[2] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. 2017. Variational Inference: A Review for Statisticians. *J. Amer. Statist. Assoc.* 112, 518 (2017), 859–877.

[3] Wen Chen, Pipei Huang, Jiaming Xu, Xin Guo, Cheng Guo, Fei Sun, Chao Li, Andreas Pfadler, Huan Zhao, and Binqiang Zhao. 2019. POG: Personalized Outfit Generation for Fashion Recommendation at Alibaba IFashion. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*. Association for Computing Machinery, 2662–2670.

[4] Xintong Han, Zuxuan Wu, Yu-Gang Jiang, and Larry S. Davis. 2017. Learning Fashion Compatibility with Bidirectional LSTMs. In *Proceedings of the 25th ACM International Conference on Multimedia (MM '17)*. New York, NY, USA, 1078–1086.

[5] Youngseung Jeon, Seungwan Jin, and Kyungsik Han. 2021. FANCY: Human-Centered, Deep Learning-Based Framework for Fashion Style Analysis. In *Proceedings of the 2021 World Wide Web Conference (WWW '21)*. 2367–2378.

[6] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR '15)*. 1–15.

[7] Tobias Kuhn, Steven Bourke, Levin Brinkmann, Tobias Buchwald, Conor Digan, Hendrik Hache, Sebastian Jaeger, Patrick Lehmann, Oskar Maier, Stefan Matting, and Yura Okulovsky. 2019. Supporting stylists by recommending fashion style. *CoRR* 1908.09493 (2019), 1–6.

[8] Jui-Hsin Lai, Bo Wu, Xin Wang, Dan Zeng, Tao Mei, and Jingen Liu. 2020. ThemeMatters: Fashion Compatibility Learning via Theme Attention. *CoRR* 1912.06227 (2020), 1–15.

[9] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (PMLR '19, Vol. 97)*. 3744–3753.

[10] Kedan Li, Chen Liu, and David Forsyth. 2019. Coherent and Controllable Outfit Generation. *CoRR* 1906.07273 (2019), 1–9.

[11] Zhi Li, Bo Wu, Qi Liu, Likang Wu, Hongke Zhao, and Tao Mei. 2020. Learning the Compositional Visual Coherence for Complementary Recommendations. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI '20)*. 3536–3543.

[12] Yen-Liang Lin, Son Tran, and Larry S. Davis. 2020. Fashion Outfit Complementary Item Retrieval. In *Proceedings of the 2020 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '20)*. 3308–3316.

[13] Zhi Lu, Yang Hu, Yan Chen, and Bing Zeng. 2021. Personalized Outfit Recommendation With Learnable Anchors. In *Proceedings of the 2021 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '21)*. 12722–12731.

[14] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Santiago, Chile) *(SIGIR '15)*. 43–52.

[15] Takuma Nakamura and Ryosuke Goto. 2018. Outfit Generation and Style Extraction via Bidirectional LSTM and Autoencoder. *CoRR* 1807.03133 (2018), 1–9.

[16] Anirudh Singhal, Ayush Chopra, Kumar Ayush, Utkarsh Patel, and Balaji Krishnamurthy. 2020. Towards a Unified Framework for Visual Compatibility Prediction. In *Proceedings of the 2020 IEEE Winter Conference on Applications of Computer Vision (WACV '2020)*. 3596–3605.

[17] Xuemeng Song, Liqiang Nie, Yinglong Wang, and Gary Marchionini. 2019. Compatibility Modeling: Data and Knowledge Applications for Clothing Matching. *Synthesis Lectures on Information Concepts, Retrieval, and Services* (2019).

[18] Mariya I. Vasileva, Bryan A. Plummer, Krishna Dusad, Shreya Rajpal, Ranjitha Kumar, and David Forsyth. 2018. Learning Type-Aware Embeddings for Fashion Compatibility. In *Proceedings of the 2018 European Conference on Computer Vision (ECCV '18)*. 405–421.

[19] Jianfeng Wang, Xiaochun Cheng, Ruomei Wang, and Shaohui Liu. 2021. Learning Outfit Compatibility with Graph Attention Network and Visual-Semantic Embedding. In *Proceedings of the 2021 IEEE International Conference on Multimedia and Expo (ICME '21)*. 1–6.

[20] Huijing Zhan, Jie Lin, Kenan Emir Ak, Boxin Shi, Ling-Yu Duan, and Alex C. Kot. 2021. A3-FKG: Attentive Attribute-Aware Fashion Knowledge Graph for Outfit Preference Prediction. *IEEE Transactions on Multimedia* (2021), 1–13.

[21] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. 2021. Dive into Deep Learning. *CoRR* 2106.11342 (2021).