eTryOn - Virtual try-ons of garments enabling novel human fashion interactions

Project Title:	**eTryOn - Virtual try-ons of garments enabling novel human fashion interactions**
Contract No:	951908 - eTryOn
Instrument:	Innovation Action
Thematic Priority:	H2020 ICT-55-2020
Start of project:	1 October 2020
Duration:	24 months

Deliverable No: 3.2

Fashion trend analysis and prediction model

Due date of deliverable:	30 November 2021
Actual submission date:	06 December 2021
Version:	Final
Main Authors:	Stefanos Papadopoulos (CERTH), Christos Koutlis (CERTH), Jamie Sutherland (MLZ), Manjunath Sudheer (MLZ), Symeon Papadopoulos (CERTH)

Project funded by the European Community under the H2020 Programme for Research and Innovation.

Deliverable title	Fashion trend analysis and prediction model
Deliverable number	3.2
Deliverable version	Final
Contractual date of delivery	30 November 2021
Actual date of delivery	06 December 2021
Deliverable filename	eTryOn_D3.2.docx
Type of deliverable	Demonstrator
Dissemination level	PU
Number of pages	46
Work package	WP3
Task(s)	T3.1, T3.2
Partner responsible	MLZ, CERTH
Author(s)	Stefanos Papadopoulos (CERTH), Christos Koutlis (CERTH), Jamie Sutherland (MLZ), Manjunath Sudheer (MLZ), Symeon Papadopoulos (CERTH)
Editor	Elisavet Chatzilari (CERTH)
Reviewer(s)	Metail

Abstract	This deliverable uses the visual features extracted from fashion imagery (using the models from D3.1) in order to build models capable of forecasting fashion trends and predict the popularity of new garment designs.

Keywords	Fashion trend forecasting, Garment popularity prediction

Copyright

Deliverable history

Version	Date	Reason	Revised by
0.1	25/10/2021	Table of Contents	Manjunath Sudheer (MLZ)
0.2	05/11/2021	Initial version	Stefanos Papadopoulos (CERTH), Christos Koutlis (CERTH), Jamie Sutherland (MLZ), Manjunath Sudheer (MLZ), Symeon Papadopoulos (CERTH)
0.3	08/11/2021	Comments by	Christos Koutlis (CERTH), Jamie Sutherland (MLZ)
0.4	09/11/2021	Pre-beta version	Stefanos Papadopoulos (CERTH), Christos Koutlis (CERTH), Jamie Sutherland (MLZ), Manjunath Sudheer (MLZ), Symeon Papadopoulos (CERTH)
0.5	19/11/2021	Further content addition and refinements (Beta version ready for internal review)	Stefanos Papadopoulos (CERTH), Christos Koutlis (CERTH), Jamie Sutherland (MLZ), Manjunath Sudheer (MLZ), Symeon Papadopoulos (CERTH)
0.6	30/11/2021	Internal review comments and suggestions by Metail	Robert Boland (Metail)
1.0	02/12/2021	Final internal version	Stefanos Papadopoulos (CERTH), Christos Koutlis (CERTH), Jamie Sutherland (MLZ), Manjunath Sudheer (MLZ), Symeon Papadopoulos (CERTH)
1.1	06/12/2021	Final version	Elisavet Chatzilari (CERTH)

List of abbreviations and Acronyms

Abbreviation	Meaning
API	Application Programming Interface
AR	Auto-regressive
AR-NN	Auto-regressive neural network
ML	Machine Learning
MLP	Multi-layer Perceptron
MPS	Mallzee Performance Score
NLP	Natural Language Processing
QAR	Quasi auto-regression
H-QAR	Hybrid quasi auto-regression
SotA	State of the art
TOPSIS	Technique for Order of Preference by Similarity to Ideal Solution

1. Executive summary

eTryon's WP3 is centered around building systems for 1) pattern recognition in fashion imagery, 2) fashion trend forecasting and garment popularity prediction and 3) fashion recommendations. The current deliverable (D3.2) uses the fashion labels and visual features extracted by models built in the framework of D3.1, in order to develop a pipeline capable of identifying fashion trends and predicting the popularity of new garment designs for different market segments, based on age and gender.

This deliverable describes in detail the research work carried out by CERTH and Mallzee, including the data collection process, the training and evaluation of machine learning models for carrying through the tasks of fashion trend forecasting and the market-segmented garment popularity prediction on new garment designs.

We proposed a hybrid quasi auto-regressive architecture, H-QAR, which combines a multi-layer perceptron utilizing the visual and categorical features of a garment and an auto-regressive network module utilizing the time series of the garment's categories and attributes. H-QAR enables inference on new garment designs based on the garment's visual appearance as well as the garment's category and attribute time series in order to compensate for the lack of historical data. Our proposed architecture was able to surpass the baseline models in both the *Mallzee* dataset and on a fashion benchmark dataset *SHIFT15m.* Moreover, our experiments on *SHIFT15m* surpassed the SotA on the task of outfit popularity prediction, and similarly, our experiments on the *Paris to Berlin* dataset were able to surpass the SotA for the task of fashion trend forecasting.

The developed models will be deployed through an API, here elaborated in Section 6, that will be used in the Designer app and as input features for the recommender systems in the Influencer and the e-commerce apps. The designers will be able to upload an image of a new garment design, the image will be passed through the "computer vision API" (from D3.1). The visual features along with the predicted category and attributes will be passed to the "popularity prediction API" which will predict the popularity score of the given garment design.

2. Introduction

Fashion is a dynamic domain, continuously changing and ever-evolving. Fashion trends and styles are highly time-dependent; they may be short-lived - quickly appear and dissipate - while others may stay relevant for years. Systematic analysis of fashion trends is not only useful for individuals who want to be up-to-date with current trends, but is also vital for fashion designers, retailers and brands. Being able to predict the longevity of fashion trends or being able to foresee emerging trends is essential for fashion brands, in order to plan their production cycles, their marketing campaigns and create products that customers will find relevant and interesting when they hit the shelves. Furthermore, fashion could also be considered a primarily visually-driven domain. Textual descriptions of products, brand awareness, available sizes, etc. may all be contributing factors in attracting customers. However, it is the visual elements, the appearance and style of the garment that are arguably one of the most important factors in attracting customer attention and leading to purchases.

Thus, including the visual aspects of fashion outfits or garments should benefit the tasks of fashion trend detection and garment popularity prediction. In recent years, this idea has been attempted and validated in a few cases thanks to the increased interest and developments in the field of computer vision. *Fashion Forward* (Al-Halah et al., 2017) is one of the first research works to extract visual features from fashion imagery and use them in order to discover fashion styles and then forecast the popularity of those styles. *GeoStyle* (Mall et al., 2019) expanded upon Fashion Forward, by collecting a large-scale image dataset from social media and forecasting fashion trends from real-world fashion imagery. More recent works have used GeoStyle to examine how fashion trends from one city may influence other cities (Al-Halah & Grauman, 2020) or how knowledge enhanced neural networks can improve trend detection (Ma H et al., 2020).

The aforementioned studies use fashion styles or attributes in order to predict fashion trends. One of their limitations is that they only work on the coarse style-level but not a finer level and can not produce accurate predictions for individual designs or outfits. They can predict, for example, that “floral dresses” will be trending next spring but all new designs of “floral dress” will receive the same prediction. To overcome this issue, Lo, L et al. (2019) collected a large-scale image dataset from lookbook.nu and trained an artificial neural network to predict the popularity of outfits. However, one noteworthy limitation is that this approach can not properly work with new fashion designs, since it requires historical data of the outfit to predict its future popularity.

In our work, we attempt to overcome both challenges and 1) perform accurate garment-level popularity predictions that can also work with 2) new garment designs; that by definition do not have historical data. To this end, we propose a hybrid quasi auto-regressive approach (H-QAR), that combines a garment-level popularity prediction module and an auto-regression neural network (AR-NN) module for fashion categories and attributes time series. H-QAR utilizes the visual features of a garment as well as the time series of this garment’s fashion categories and attributes. Our rationale was that while new garment designs, by definition, do not have historical data, they would still belong to a particular fashion category and have multiple attributes and that modelling those time series would be valuable and informative proxies of popularity.

We experimented with 5 AR-NN models in Mallzee's dataset and on a publicly available dataset, *Paris to Berlin* (Al-Halah & Grauman, 2020). On the latter, our Feedback LSTM model was able to surpass the SotA by lowering the MAE down to 0.0675 and MAPE to 16.67 compared to 0.0699 and 17.38 scores respectively. Moreover, we applied H-QAR on two datasets, one collected by project partners Mallzee and one publicly available fashion dataset: *SHIFT15m* (Kimura et al., 2021). Compared against a baseline linear regression model and a multilayer perceptron regressor (MLP Regressor), H-QAR performed better in both datasets.

Despite the enhanced performance of H-QAR, it could only predict a general popularity score without taking any demographic information into account. A new design of a "floral dress" would have the same popularity score for "females, age: 18-25" with "males, age: 40-50". Such predictions would have limited usefulness for designers, retailers and marketing campaign strategists. To alleviate this issue and perform more informed and useful predictions for different segments of the market, we expanded the existing Mallzee dataset to include the daily popularity scores for different age and gender groups. Again, H-QAR was able to outperform both the baseline linear regression model and the MLP Regressor on Mallzee's market segmentation dataset.

2.1 Technical framework

This deliverable is a continuation of the previous deliverable, D3.1: *Pattern Recognition on Fashion Imagery*. Our central objective in D3.1 was to build deep learning models able to automatically recognize patterns in fashion imagery and classify garments into categories and detect their fine-grained attributes. To this end, we developed a hierarchical deep learning pipeline consisting of three stages illustrated in Figure 2.1. First, fashion imagery is passed through an object-type detection model that identifies the location of various garments in an image and classifies them into high-level classes (upper body, lower body, full body and footwear). The image is then cropped around the predicted bounding boxes of the garments and the cropped images are passed to the second and third stage, consisting of two classifiers, one category-level and one attribute-level. This approach enables the analysis of fashion imagery depicting complete outfits.

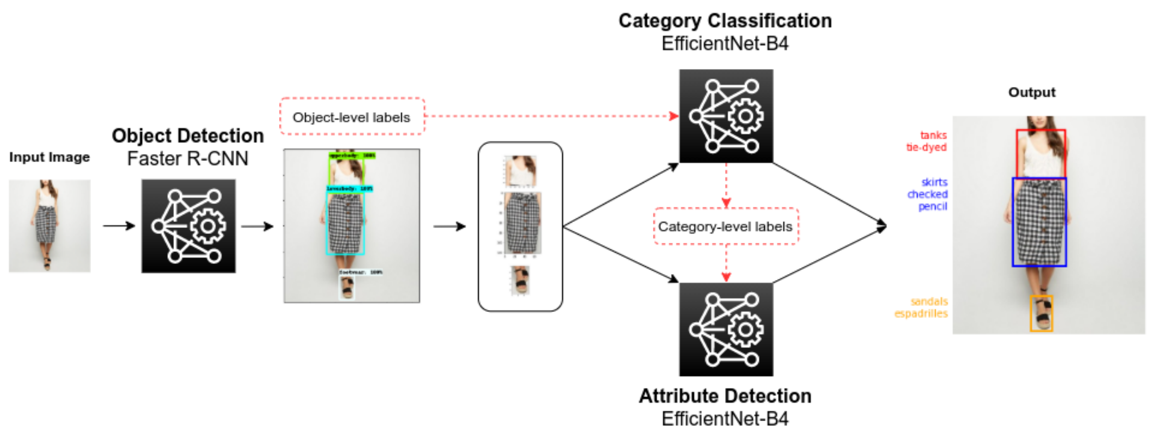


Figure 2.1: Workflow of the hierarchical deep learning pipeline for image pattern recognition from deliverable D3.1.

For our work in D3.2, we utilized the hierarchical pipeline (from D3.1) in order to extract fashion labels and visual features from fashion imagery and then use them for the task of predicting garment popularity.

2.2 Applications within eTryOn

eTryOn aims to develop three applications focusing on virtual try-ons of clothing. All these applications will require input from the present deliverable, namely:

- The VR designer app, targeting fashion designers interested in designing and evaluating new garments. Our work from this deliverable will provide predictions about how popular a certain design will be in the market at a specific point in the future. Additionally we will offer informed predictions for different segments of the market, namely different genders and age groups.
- The DressMeUp app, targeting social media users (influencers) who will receive market-segmented popularity predictions similar to the VR designer app.
- The magic mirror app, targeting fashion consumers (ecommerce), where the popularity scores produced by the methods described in this deliverable will inform the underlying recommendation engine.

This deliverable will provide the necessary steps of training and evaluating machine learning models to identify fashion trends of fashion categories and attributes and to predict the popularity of newly designed garments. The developed models will then be available via API endpoints for use in the eTryOn applications.

2.3 Structure of the report

The teams involved in this deliverable (Mallzee and CERTH) have devoted effort collecting and cleaning large-scale fashion datasets from Mallzee's databases as well as training and evaluating machine learning models for fashion trend forecasting and garment popularity prediction.

- Section 3 describes the process of collecting and quality checking novel fashion datasets from Mallzee's dataset. Moreover it offers an exploratory analysis of these datasets along with two publicly available datasets that were used for further testing and validating our findings.

- Section 4 discusses the development of various machine learning models for fashion trend forecasting and garment popularity prediction.
- Section 5 presents the evaluation protocol, the experimental setup of each task as well as the empirical results of those experiments.
- Section 6 describes our work on integrating the trained machine learning models (from both D3.1 and D3.2) into API endpoints that will be used in the eTryOn applications.
- Section 7 concludes the deliverable with an overview of our work and the authors' final remarks.

3. Data collection, processing and exploration

We had to create large-scale datasets consisting of historical interactions between users and products in order to train machine learning models capable of detecting fashion trends and predicting the popularity of garments. We had to define positive and negative metrics for differentiating those interactions and assess the popularity of a given garment at a specific point in time. To this end, we collected and processed a large dataset from Mallzee and defined the Mallzee Performance Score (MPS), which takes into account both positive and negative types of interactions. Furthermore, we experimented with two publicly available datasets: *Paris to Berlin* and *SHIFT15m* in order to compare our methods against the SotA. The data collection and cleaning processes are presented in this section alongside an exploratory analysis of each dataset.

3.1 Mallzee datasets

We have created three separate datasets using the raw daily data from Mallzee's databases, namely the: 1) *Mallzee MPS dataset* used for garment popularity prediction, 2) *Mallzee market segmentation* dataset used for garment popularity prediction based on user demographics (age and gender) and 3) the *Mallzee time series* dataset for detecting trends in fashion categories and attributes. All three datasets use the *Mallzee performance score* (MPS) as the target variable.

3.1.1 Mallzee performance score (MPS)

The Mallzee Performance Score (MPS) is a metric assessing how much a product is appreciated at a specific point in time within the Mallzee app and is used by Mallzee for internal analysis and decision making. The metric uses the historical user-to-product interactions and takes into account all the different contexts in the Mallzee app. MPS measures the fraction of total positive interactions (*saves*, *additions to bags*) a product has received out of the total times it has been seen; which includes both positive and negative interactions (*hides*). It is calculated as below:

$$MPS = Positive\ Interactions\ /\ (Positive\ Interactions\ +\ Negative\ Interactions)\ *\ 100$$

It also takes into account the mode that the interactions took place (whether *swipe*, *grid* modes) and weighs the interactions accordingly. In all following Mallzee datasets, we normalize the MPS within a range of (0,1) with the use of min-max scaling. The scaler is fitted on the training data and is then applied on the validation and testing sets. This setting ensures that no information regarding the evaluation sets will be 'leaked' during the training process of the machine learning models.

3.1.2 Mallzee MPS dataset

Mallzee collected all interactions between users and products from their database and aggregated the number of positive and negative interactions that each product received in a day. This process resulted in 256,396,168 records of aggregated daily interactions between 2017 and 2021. However, the datasets contained multiple duplicate records and had numerous instances with zero *positive interactions*. The latter issue would lead to an extremely sparse dataset with mostly unpopular products. We filtered the whole dataset based on the following criteria: 1) we kept records with more than zero *positive interactions* and 2) dropped duplicate records based on the product id and date; thus keeping only one daily record per product. The filtering process resulted in 6,020,528 records regarding 743,323 unique products. The data span 1081 days between 2017-10-16 and 2020-09-30. Previous and later time periods were filtered out for not having any positive interactions.

Thereafter, we had to classify each garment's category and attributes in order to identify trends in those categories and attributes. One central objective of our work was to examine how image features can contribute in the fashion trend detection and garment popularity prediction tasks. To this end, we had to collect the images of the products in order to extract their *category*, *attributes* and *visual features*, and we developed a real-time inference pipeline. The pipeline would retrieve images from URLs either from Mallzee's website or their collaborating retailers for all unique product_ids. Then, each image was passed through the three image processing models developed during D3.1, namely: 1) object-type detector, 2) category classifier and 3) attribute classifier. This pipeline extracted the labels from all three stages and also extracted and saved their visual features, which were extracted from the last convolutional layers of the category and attribute classifiers.

We had to filter out images with broken URLs and products of which predicted object types did not match the product's ground truth object type; which was available through Mallzee's database. If for example a product was annotated as 'full-body' but the object detection model predicted an 'upper body' and a 'lower body' garment separately, this product would be filtered out. This consistency check ensured that both wrong annotations and wrong predictions would be filtered out; thus largely reducing noise in the dataset. This was based on the assumption that if the pipeline can not correctly predict the object type of the garment then the categories and attributes could likely lead to mistaken predictions. For example, if a product's image depicted a "dress" but the model mistakenly predicted two separate garments, a "blouse" and a "skirt", this would create a mismatch between the product's actual and predicted labels; since the pipeline would crop the bounding boxes around the predicted "blouse" and "skirt" and the classifiers would not be able to predict the correct label. At the same time, this consistency check would filter out some cases of mistaken object type annotations; which were extracted with NLP methods by the product's textual descriptions and thus could entail some errors.

From the initial count of 743,736 images, 678,496 were scraped successfully and from those 571,333 had matching ground truth and predicted object type. Merging the 571,333 successfully inferenced product images with the *total interactions* file, reduced the total number of interactions to 4,767,445, which is still a sufficient amount of data for training machine learning models. We refer to this dataset as the *Mallzee MPS dataset* and used it for training *garment popularity prediction* models. For each record, the *Mallzee MPS dataset* contains multiple types of features including: categorical (color, category and attribute labels), numerical (average price), visual (category-level and attribute-level features) and temporal features (day, week, month). The MPS - which is also numerical - is defined as the target variable thus rendering the task of garment popularity prediction as a regression task.

We randomly shuffled the dataset and then split it with 0.8, 0.1 and 0.1 ratios for training, validation and testing respectively. We used the exact same data splits for all experiments in order to ensure comparability across different experiments.

3.1.3 Mallzee market segmentation dataset

Models trained on the *Mallzee MPS* dataset will learn to predict the general popularity of a garment. Those predictions will be identical regardless of the demographic group (age group and gender). A new design of a “floral dress” would have the same popularity score for “females, age: 18-25” with “males, age: 40-50”. In order to alleviate this issue and perform more informed and useful predictions for different segments of the market, we expanded the existing Mallzee dataset to include the daily popularity scores for different age and gender groups.

Mallzee collected all available data regarding the daily MPS per demographic group. The demographics consist of two gender groups (males, females) and 13 age-groups including: 0-18, 18-25, 25-30, 30-35, 35-40, 40-45, 45-50, 50-55, 55-60, 60-65, 65-70, 70-75, 75-80. We combined all age groups with the two genders, thus creating 26 demographic combinations in total, for example “males 0-18” and “females 30-35”. The raw dataset included 60,978,519 records with the product-id, date, age-group, gender and MPS. These raw records were centered around the 571,333 unique products from the Mallzee MPS dataset across multiple demographic groups and dates. Each product could have multiple records for different dates and for the various age and gender groups. Moreover, each record would have its own popularity score; essentially showing how different demographic groups reacted to a specific product at a given date. We combined the *Mallzee MPS* data with the raw demographics data on the product-id and date; resulting in 12,188,276 records. However, 57% of the records’ popularity scores were equal to zero. A popularity score of zero could be the result of genuine dislike for said product for a particular demographic or simply the result of having too few interactions between a demographic group and a particular product (if those few interactions happen to be negative). In order to minimize this possibility and reduce the “noise” of the dataset, we only kept the zero scores from the “most seen” products; those products that received the most attention (either positive and negative in general). Our assumption was that it is more likely that the demographic group in question would have interacted with the “most seen” products and that the zero MPS score would reflect genuine dislike. Our reasoning for this filtering operation was that it would be difficult to train a ML model on data that consisted of approximately 57% zero target values. This assumption was empirically validated. The model trained on the filtered dataset reached a mean absolute error (MAE) of 0.22 (as seen in Table 5.4-A) while the model trained on the unfiltered dataset scored 0.27; which translates into a higher error rate. Moreover the distribution of the predicted values were vastly different from the ground truth values; which consisted of 57% zeros. More specifically, we chose a percentile of 85.6% because it produced 259,124 zero MPS records that approximately equal to 5% of the non-zero dataset (257,653). The final dataset contains 5,412,193 records. Similarly to the *Mallzee MPS* dataset we split the *market segmentation* dataset into training, validation and testing sets with 0.8, 0.1, 0.1 ratios respectively.

3.1.4 Mallzee time series dataset

In order to create a time series dataset for fashion categories and attributes, we used the Mallzee MPS dataset and applied the following operations.

- Sort the data by date in ascending order.
- Group the data by *date* and *category* (or *attribute*).
- Aggregate the mean MPS (per day per category).
- Pivot the data-frame: *dates* become the index, *categories* become the columns and the *MPS* become the values of the matrix.

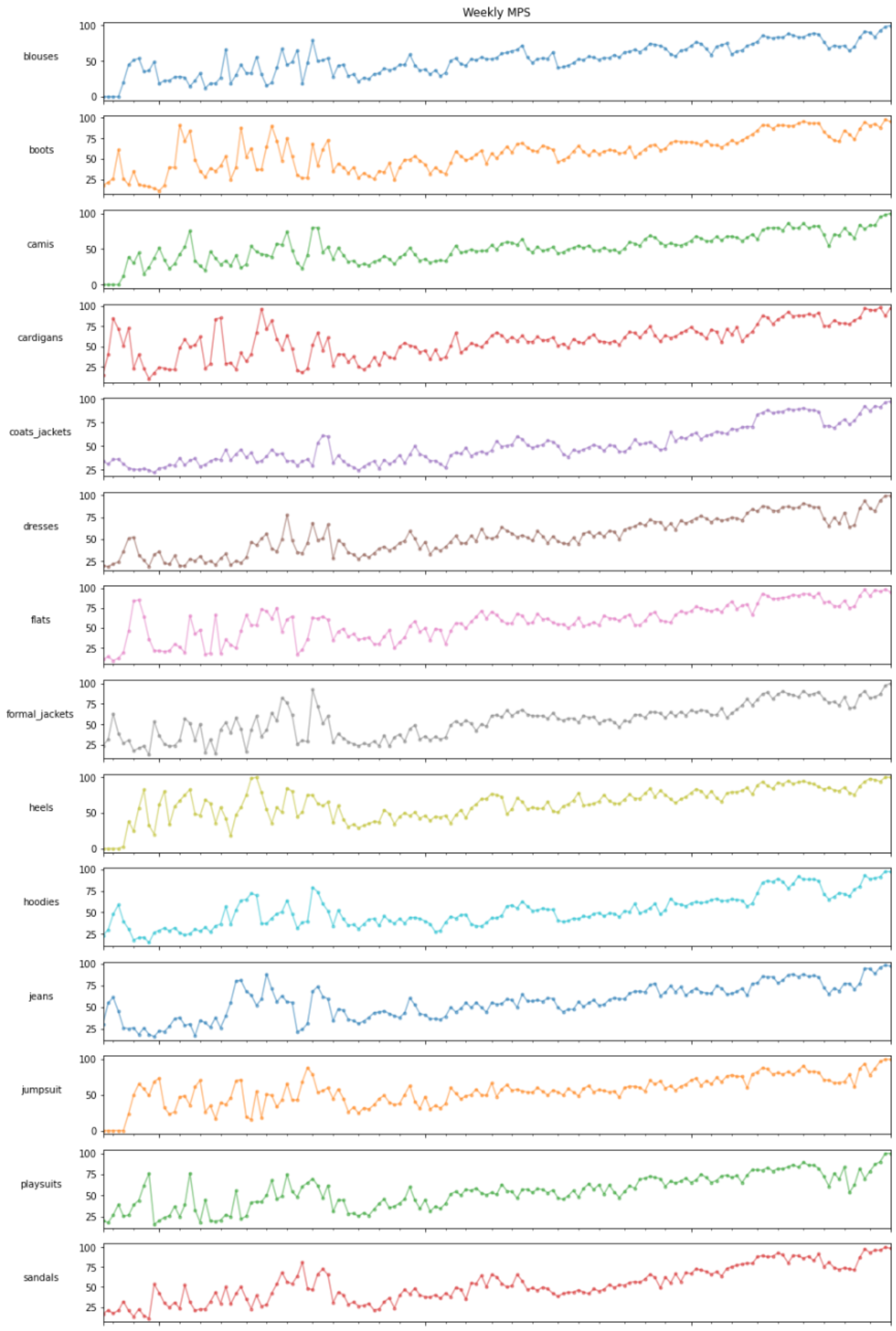
When calculating the MPS in weekly aggregations, we do not directly aggregate the daily mean MPS values. Rather, we calculate the sum of weekly positive and sum of weekly negative interactions for each category and attribute and then calculate the MPS as the weekly-positive-sum / (weekly-positive-sum + weekly-negative-sum). We similarly calculate the monthly MPS by summing the monthly positive and monthly negative interactions.

The resulting dataset consists of 22 category time series and 109 attribute time series spanning 1081 days, or 155 weeks, or 36 months. We calculated the percentage of missing values in order to assess the sparsity of the time series. Category time series had 5.26% daily sparsity, 0.88% weekly sparsity and 0.50% monthly sparsity. Attributes time series had 22.04% daily sparsity, 8.92% weekly sparsity and 5.48% monthly sparsity. These levels of sparsity can already be considered relatively low; however, with the use of linear interpolation we eliminated all missing values.

From all available time series, we indicatively present the weekly aggregated time series for the 22 categories in Figure 3.1.4-1 as well as the overall weekly aggregated time series in Figure 3.1.4-2; where we can observe the general trend of the data**.** We can also observe certain irregularities with the time series of fashion categories. Firstly, all category time series seem to follow an upward trend that peaks at the later months of 2020. This increase seemingly coincides with the outbreak of COVID-19. However, the MPS is a normalised metric between positive and negative interactions. An increase in activity could not explain the increase in positive interactions - hence the increase in MPS - on its own. During the same period, Mallzee has made improvements in their recommendation system. These changes have led to improved brand targeting and personalisation. As a result, users of the Mallzee app received more relevant recommendations and thus the overall fraction of positive interactions has increased, leading to an upward trend in all garment categories during the later months of 2020.

Secondly, there is a lack of seasonal variation and trends. Intuitively, we would expect, for example, increased interest during the fall and winter for sweaters and jackets and decreased during summer. No such clear seasonal trends can be found in the Mallzee time series dataset. One possible explanation could be the lack of historical purchase data. As mentioned in Section 3.1.1, the positive interactions that make up MPS consist of *views*, *saves* and *additions to the (shopping) bag*. Direct sales are not available in the

Mallzee datasets, since Mallzee mainly re-directs their users to the retailers' websites. Moreover, the *additions to the bag* were very few. Namely, there are 49,465 interactions out of the approximately 5 million total interactions in the *Mallzee MPS* dataset. This means that MPS mainly indicates likability but not necessarily intention of purchase. This means that users may still interact positively with 'out-of-season' garments (e.g a leather jacket during the summer) but they would not proceed to a purchase.



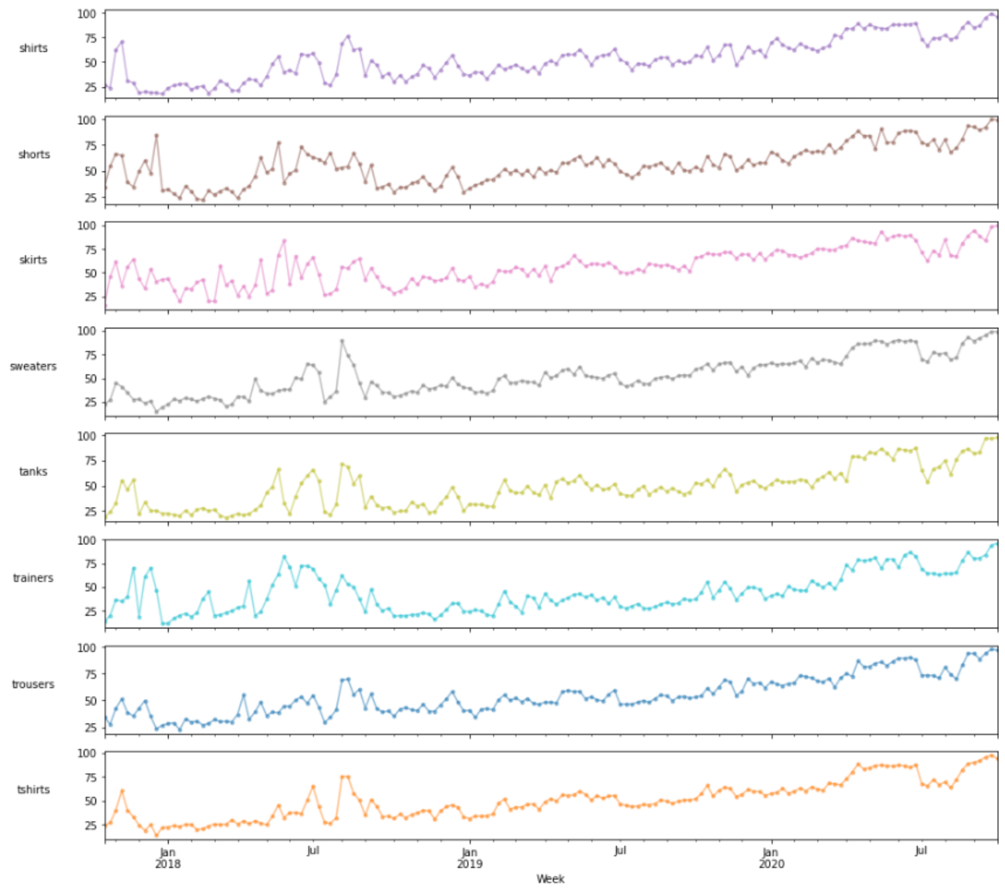


Figure 3.1.4-1: Weekly aggregated time series for the 22 fashion categories in the Mallzee MPS dataset.

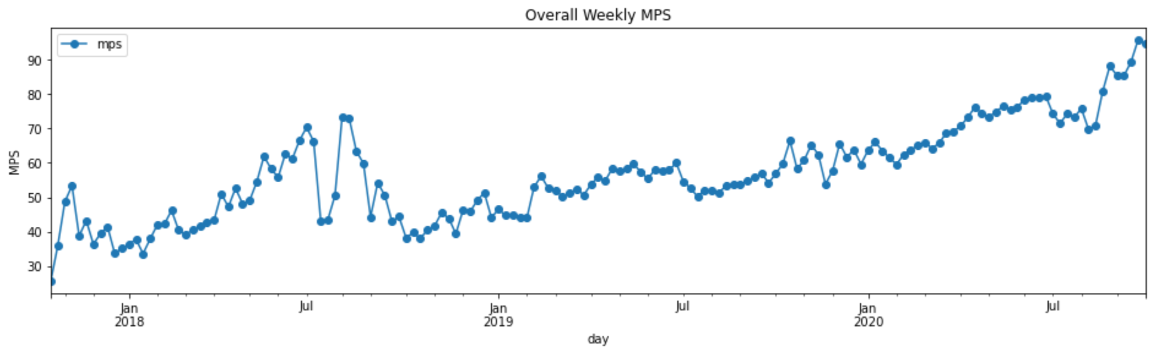


Figure 3.1.4-2 Weekly aggregated time series for the overall Mallzee MPS dataset.

3.2 Benchmark datasets

To more thoroughly examine the performance of our models and validate our findings we applied our methods on publicly available benchmark datasets. To this end, we utilize two datasets, one for benchmarking fashion trend forecasting of fashion categories and attributes, and one for benchmarking the prediction of garments' popularity based on their visual features.

3.2.1 Paris to Berlin

We consider the *GeoStyle dataset* which is a widely-used dataset for benchmarking fashion trends based on visual features extracted from street fashion images collected from social media platforms (Mall et al., 2019). However, the dataset is no longer publicly available. Instead, only a processed version of the dataset is still available by Al-Halah & Grauman (2020) who used a GoogLeNet model trained on fashion imagery in order to classify the GeoStyle images into 46 fashion attributes. Thereafter, they inferred the popularity of each of the 46 attributes for each city in the GeoStyle dataset which contains 50 cities in total. The Paris to Berlin dataset consists of 149 weekly instances between 2013 and 2016. The data are regarding the popularity of 46 (anonymised) fashion attributes in 50 different cities from around the world; resulting in 2200 time series in total. Given the structure of the *Paris to Berlin* dataset we could only use it for fashion trend forecasting and not for garment popularity prediction.

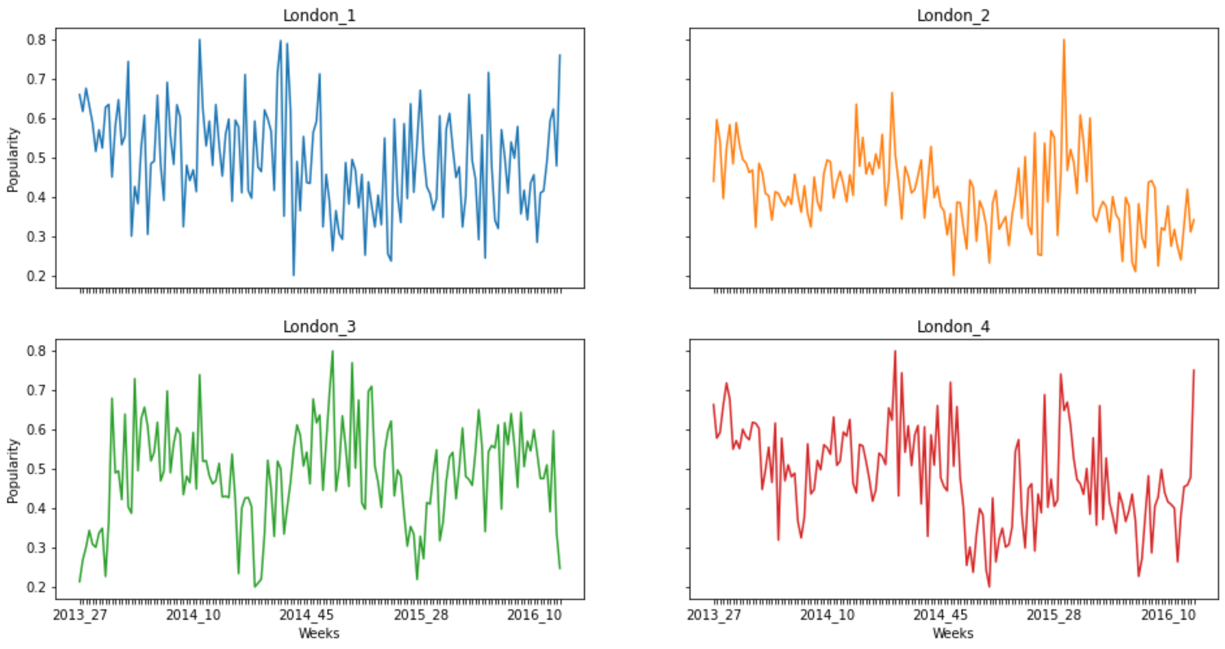


Figure 3.2.1: Time series representing the popularity of four fashion attributes in the city of London. Using the *Paris to Berlin* dataset.

3.2.2 SHIFT15m

Since *Paris to Berlin* can only be used for trend forecasting of fashion attributes, we also consider another dataset for comparing our models' performance on the *garment popularity prediction* task. Recently, Lo et al. (2019) collected a large-scale image dataset from lookbook.nu for evaluating the popularity of fashion outfits. However, the authors did not offer their collected dataset publicly. As an alternative, we ended up using the newly published SHIFT15m dataset (Kimura et al., 2021). SHIFT15m is a large-scale multi-objective fashion dataset that can be used, among other tasks, for outfit popularity prediction. SHIFT15m contains 15 million instances of outfits from 2.5 million unique garments. Each outfit instance has a number of likes that the outfit received in a particular day. The number of likes is defined as the target variable. For our purposes -

which is the garment-level popularity prediction - we had to split each outfit into its individual garments in order to perform garment popularity prediction. However, as will be discussed in section 5.3.2, we also performed experiments with the complete outfits in order to ensure the comparability of our models.

To this end, we remap *SHIFT15m* from the initial outfit-level popularity onto the garment-level. We split each outfit into its component garments and define the outfit's received number of likes onto each garment. For each garment we use visual features given by SHIFT15m, which were extracted through a VGG16 image network pre-trained on the ImageNet dataset. Due to limitations in computational resources, we could not use all 15 million outfit instances. Therefore, we used two versions of the dataset. One between 2017-2020 and one between 2016-2020; while the total range is between 2010 and 2020. This time range matches a similar time span to the one found in the *Mallzee MPS* dataset. We filter out all instances outside this time range and merge the remaining products with their corresponding visual features. The filtered garment-level SHIFT-2016-2020 dataset consists of 3,742,492 records regarding 1,038,107 unique products. While the SHIFT-2017-2020 range has 1,888,845 instances consisting of 712,108 unique garments. We use two versions of the same dataset in order to assess how different scales of data may affect the performance of the models.

Our two versions of SHIFT15m consist of 6 columns: 1) Item id, 2) date, 3) category ID1, 4) category ID2, 5) visual features and 6) the number of likes. Category ID1 and ID2 are fashion classifications similar to Mallzee's 'categories' and 'attributes'. ID1 contains 7 unique values while ID2 contains 43 unique values. The categories are provided as encoded numerical values and their actual names are not known. Each row in the 'visual features' column contains an array of size 4096. As can be seen in Figure 3.2.2, the number of likes had a highly imbalanced distribution. In order to bring the target variable closer to a normal distribution we normalised the number of likes with the logarithmic transformation:

$$Normalized\ likes = log(likes + 1)$$

Then, we re-normalized the logarithmic number of likes within the range of 0 and 1 with the use of min-max scaling.

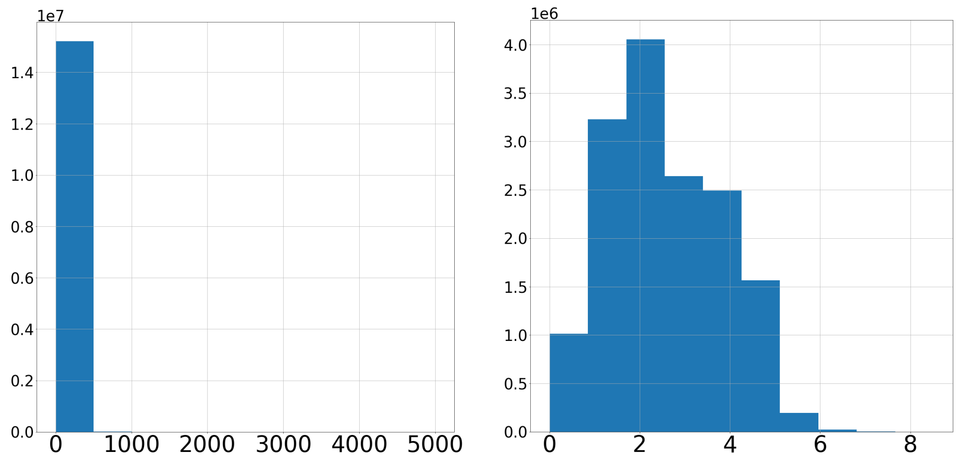


Figure 3.2.2 Initial distribution of the number of likes in *SHIFT15m* (left) and the normalised distribution based on the logarithmic transformation (right).

Finally, as with the *Mallzee MPS* dataset, we randomly shuffled *SHIFT15m* and then split it into training, validation and testing sets with 0.8, 0.1, 0.1 rations respectively. All experiments used the exact same splits.

4. Modelling fashion trends and garment popularity prediction

4.1 Fashion trends forecasting

For the task of fashion trend forecasting, we developed and trained multiple auto-regressive (AR) neural networks (NN) trend forecasting models on the *Mallzee time series* and the *Paris to Berlin* time series datasets. We did not utilize any auxiliary features, but rather, we sorted the data by dates in an ascending order, grouped by category (or attributes) and day and defined the popularity score as the target variable.

We pre-process the data so as to create windowed time steps. For each experiment, we had to define the number of input steps and the number of output steps. Meaning, how many *IN-STEP* days the model will receive as input in order to predict *OUT-STEP* days into the future. An AR model is considered single-step if it has to predict only a single time step in the future (an example is shown in the left graph of Figure 4.1-1). Conversely, a model is considered multi-step if it has to predict multiple time steps in the future (an example is shown in the right graph of Figure 4.1-1).

Moreover, we trained multi-target AR-NN models. This means that each model received multivariate and not univariate time series. We assumed that fashion categories and attributes are not independent among each other. Increased interest in one fashion category may result in reduced interest in another category while others may have positively correlated trajectories.

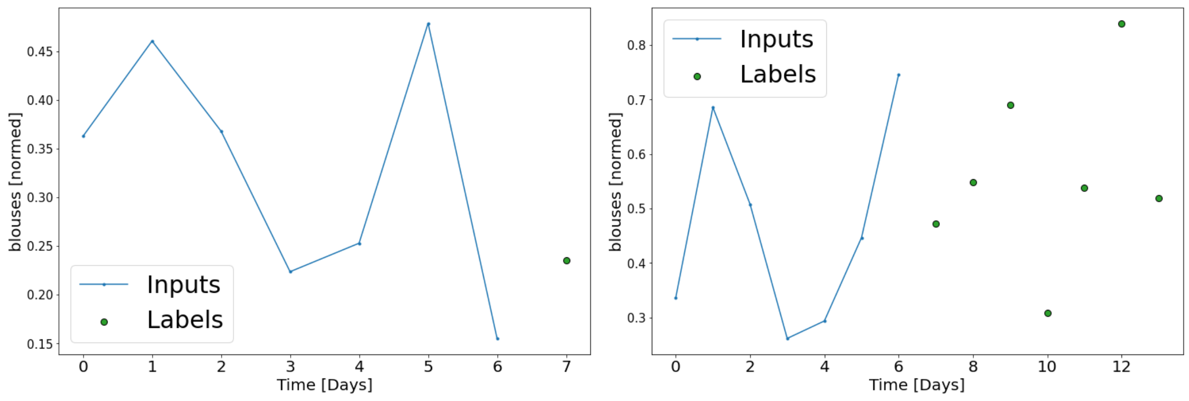


Figure 4.1-1 Examples of single-step time series (left) and multi-steps time series (right).

We selected to experiment with the following AR-NN models, since they could satisfy the criteria of being both multi-target and multi-step and that have been used in numerous works for time series forecasting (Lim & Zohren, 2021):

- Long Short Term Memory RNN (LSTM) (Hochreiter & Schmidhuber, 1997)
- Convolutional Neural Network (CNN) (Zhao et al, 2017)
- Convolutional LSTM (ConvLSTM) (Xingjian et al., 2015)
- Transformer (Vaswani et al., 2017)
- Feedback LSTM (Graves, 2013)

All models receive a time series of shape: (BATCH-SIZE, IN-STEPS, NUMBER OF FEATURES) as input and return an array of (BATCH-SIZE, OUT-STEPS, NUM OF FEATURES) as output. The batch-size and the IN/OUT steps are hyper-parameters defined before each experiment. The number of features is equal to 22 for categories and 109 for attributes for the *Mallzee time series* dataset and 2200 for *Paris to Berlin*. A linear layer (prediction layer) is added on top of all architectures with (OUT-STEPS * NUM-OF-FEATURES) units who are then reshaped into (BATCH-SIZE, OUT-STEPS, NUM OF FEATURES) in order to predict the outcome.

Long-short term memory (LSTM) is a type of recurrent neural network (RNN) that is well-suited for time series data due to processing time series step-by-step, maintaining an internal state from one step to the next. We used the original LSTM architecture, as proposed by Hochreiter & Schmidhuber (1997). A single LSTM cell is shown in Figure 4.1-2. For LSTMs we defined the number of LSTM layers and the number of LSTM units for each layer. In between LSTM layers, we use a dropout layer of 10% probability in order to reduce overfitting.

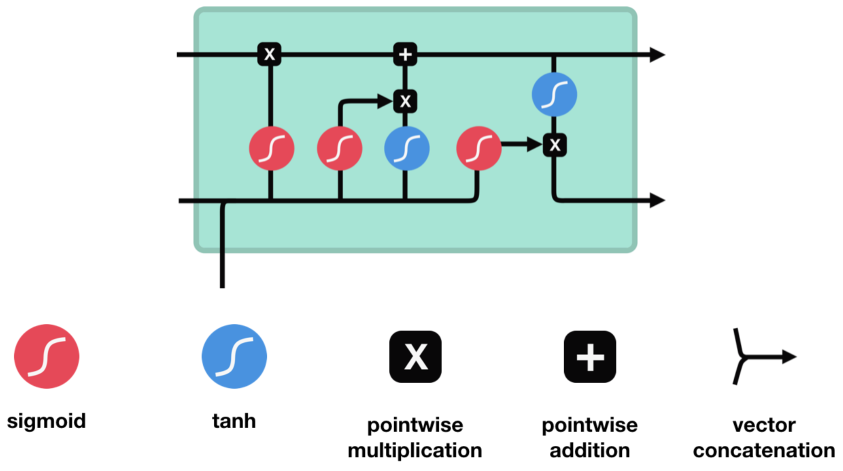


Figure 4.1-2: A Long Short-Term Memory (LSTM) cell processes data sequentially and maintains its hidden state through time. Source: Illustrated Guide to LSTM's and GRU's[1]

For the *convolutional neural network* (CNN), we stacked one-dimensional convolutional layers for which we define the number of filters as well as the size for the kernel that passes over the data. Figure 4.1-3 presents the convolutional process in a one-dimensional CNN. Between the convolutional layers we use a dropout layer of 10% probability in order to reduce overfitting and also experimented with adding a batch normalisation layer before the dropout. A similar architecture was used by Zhao et al. (2017) for time series classification. The difference is that on top of the last convolutional layer we add a global average pooling layer whose output is given to the linear prediction layer.

As the name implies, *convolutional long-short term memory* (ConvLSTM) is the combination of the last two architectures and was proposed by Xingjian et al. (2015). ConvLSTM uses a number of one-dimensional conv layers and on top of them, a

[1] Illustrated Guide to LSTM's and GRU's: A step by step explanation
towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

number of LSTM layers. For each experiment we define 1) the number of conv layers, 2) the size of the conv filters, 3) the size of the convolutional kernel, as well as 4) the number of LSTM layers and 5) the amount of LSTM units.

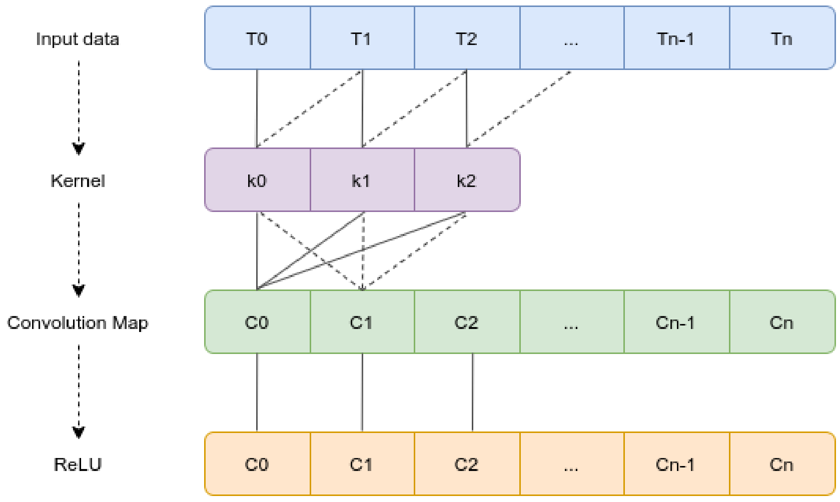


Figure 4.1-3: One-dimensional convolutional neural network.

For the *Transformer*, we created transformer encoder blocks that make use of multi-head attention layers. The number of attention heads and their embedding size were hyper-parameters that required tuning. Each transformer block uses layer normalisation, residual connections and dropout, similarly to Vaswani et al. (2017). The number of transformer blocks was also a hyper-parameter that required tuning.

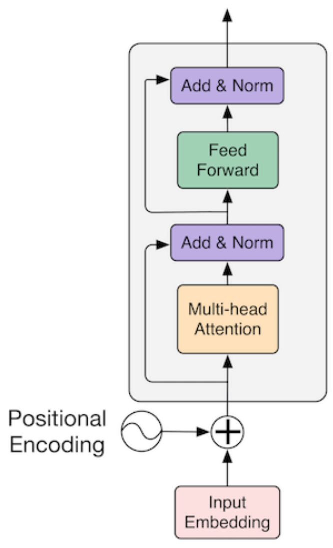


Figure 4.1-4: Structure of a single transformer encoder block. Source: Ghader (2021)

All aforementioned models predict the entire output sequence in a single step but *Feedback LSTM* - proposed by Graves (2013) - decomposes the prediction into individual time steps. Each model's output is fed back into itself at each step. Therefore the model’s predictions can be made conditional on the previous step. Feedback LSTM requires a warm-up method that receives the whole time series in order to initialize the internal state and capture the relevant parts of the input history. The warmup method uses a number of LSTM layers and a final single-step prediction layer that returns a single prediction and a hidden state. Thereafter, for each step in IN-steps, the previous prediction and the previous hidden state is fed into a LSTM Cell and a prediction layer predicts the next step. All single-step predictions are collected, stacked and returned as the output of the model. For this architecture we had to define 1) the number of LSTM layers, 2) the size of their units and 3) if another dense layer (activated by 'RelU') will be added before the final prediction layer.

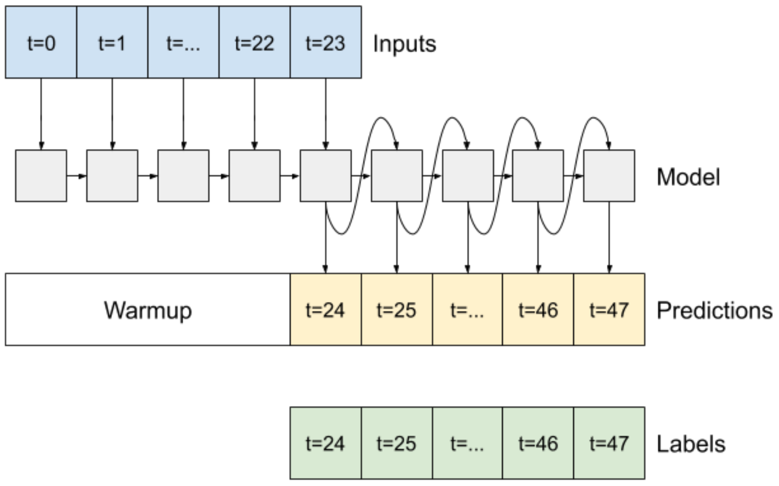


Figure 4.1-5: Feedback LSTM decomposes multi-step predictions into individual time steps and uses the previous prediction as input for predicting the next step. Source: Time Series Forecasting (TensorFlow)[2]

4.2 Garment popularity prediction

One significant limitation of the trend forecasting models for fashion categories and attributes is that they can not offer specialized predictions for new products; since all products of the same category and attributes will receive the same prediction for a specific point in time. Therefore, we considered training machine learning models to predict the popularity of individual garments, including 'cold' (previously unseen) products, based on their features. We examine two approaches: a conventional regression setting (baseline model) and a quasi auto-regressive setting (H-QAR). The latter combines the regression module with the AR-NN trend forecasting models of the previous section. Furthermore, we examine how different features affect the performance of the *baseline regression* setting and how the two settings compare with each other.

4.2.1 MLP Regressor

We develop a multi-layer perceptron regression neural network (MLP Regressor) for predicting the popularity of fashion garments. Our MLP Regressor can receive different types of features including categorical (colors, category and attribute labels), numerical (average price), visual features (extracted by computer vision models developed in D3.1) and temporal features (day, week and month). The network is modular, meaning that the researcher can select which features to include for a particular experiment. Numerical

[2] www.tensorflow.org/tutorials/structured_data/time_series#advanced_autoregressive_model

features (the average price and the MPS) are normalised within a range of (0,1) with the use of min-max scaling. Each min-max scaler was first ‘fitted’ on the training data and then applied on the validation and testing sets as well. On the other hand, categorical features are encoded with numerical values and given to separate embedding layers which map positive integers (encoded labels or indexes) into dense vectors of fixed size. Temporal features are not treated as in the fashion trend forecasting task but rather are encoded and given to separate embedding layers, similarly to the categorical features. Finally, we applied in-batch normalisation visual features with the use of the L2 norm.

We created a custom data generator that loads batches of data from the dataframe, shuffles the training batches (but not the validation and testing data) and performs all aforementioned pre-processing operations. The data generator returns only those features that were selected for a particular experiment. Thereafter, categorical and temporal features are passed through different embedding layers and their outputs are concatenated with the rest of features (numeric and visual). Then, all features are passed through N dense layers of U units (activated by the ReLU function) and finally, a prediction layer (outputs a single value) predicting the popularity score of that garment. N and U are hyper-parameters that require tuning. The prediction layer uses a linear activation function but we also experimented with ReLU and sigmoid activation functions. The network is optimised by an Adam optimizer with the use of the MSE (mean squared error) loss function which calculates the difference between the predicted and the ground truth popularity scores and optimises the network’s weights accordingly. We also experimented with LogCosh (logarithm of the hyperbolic cosine) and MAE (mean absolute error) for the loss function. The workflow of the MLP Regressor can be seen in Figure 4.2.1-1.

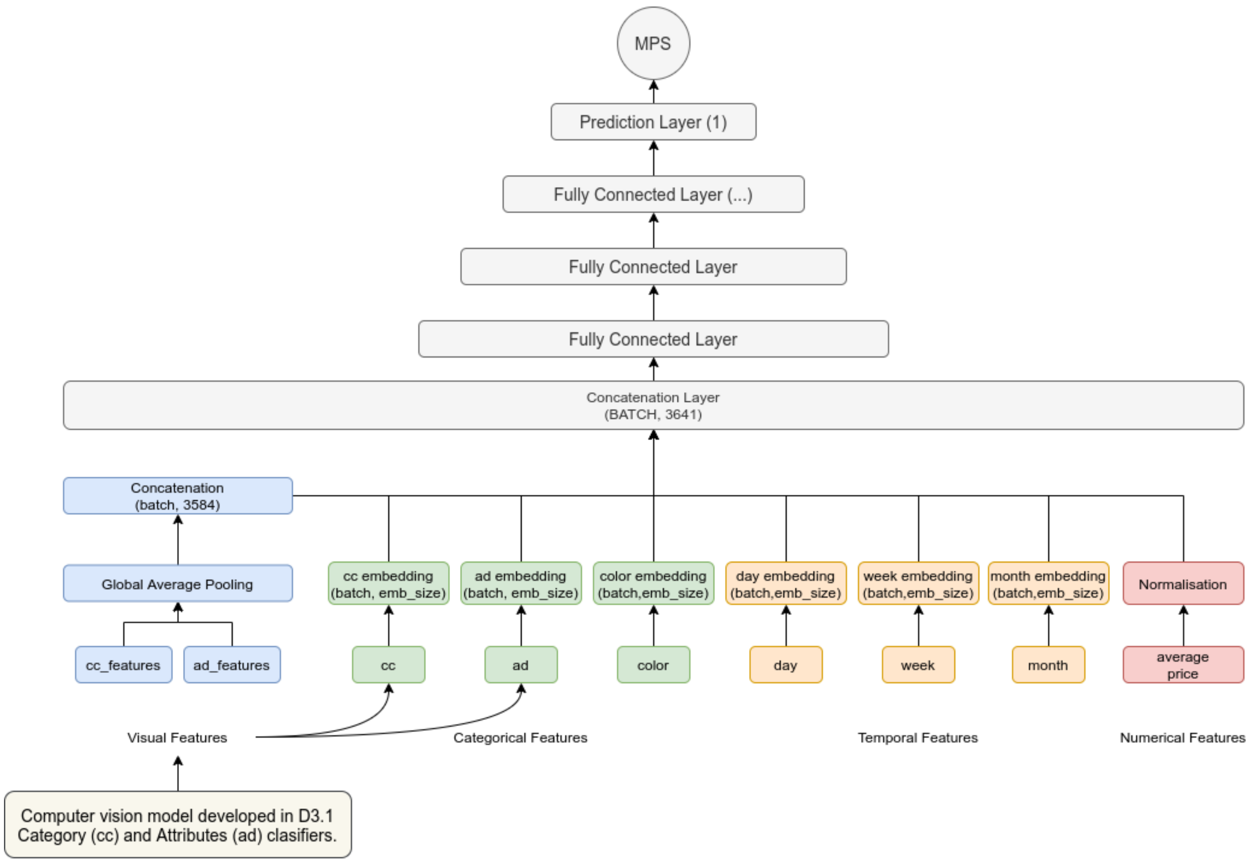

MPS
Prediction Layer (1)
Fully Connected Layer (...)
Fully Connected Layer
Fully Connected Layer
Concatenation Layer (BATCH, 3641)
Concatenation (batch, 3584)
Global Average Pooling
cc_features
ad_features
cc embedding (batch, emb_size)
ad embedding (batch, emb_size)
color embedding (batch,emb_size)
day embedding (batch,emb_size)
week embedding (batch,emb_size)
month embedding (batch,emb_size)
Normalisation
cc
ad
color
day
week
month
average price
Visual Features
Categorical Features
Temporal Features
Numerical Features
Computer vision model developed in D3.1 Category (cc) and Attributes (ad) clasifiers.

Figure 4.2.1-1: Workflow of the *MLP Regressor* (baseline model).

4.2.2 Hybrid quasi auto-regression (H-QAR)

The AR-NN models can solely predict the popularity of fashion categories and attributes but not the popularity of individual garments. On the other hand, the MLP Regressor is able to perform predictions on individual garments (even if no historical data are available for these garments), but it does not take into consideration historical fashion trends. We hypothesize that a model combining both settings (visual features and fashion trends) would yield improved predictive performance. Our hypothesis is that historical data of a particular garment are not always available, especially for new products, and that the time series of the garment’s categories and attributes could be useful popularity proxies. To test this hypothesis, we developed *H-QAR*, which utilizes both the MLP Regressor and AR-NN models.

The regression module utilizes the visual features and the garment’s category and attributes. The auto-regressive module receives *IN-STEPS* past time series of those categories and attributes. We describe the process as *quasi auto-regressive* (QAR) because *H-QAR* receives the time series of the garment’s category and attributes; not the past time series of the garment itself.

We used the data generator from the MLP Regressor but expanded it to fetch the time series of the garment's category and attributes for the past *IN-STEPS* popularity scores before the current date that we try to predict.

For the AR-NN module, we experimented with the same 5 architectures discussed in Section 4.1 (LSTM, CNN, ConvLSTM, Feedback LSTM, Transformer) but we have removed their top prediction layers. Instead, we extract the features from their last AR-NN layer and then, we concatenate them with the features from the last fully connected layer of the regression module. The workflow of the *H-QAR* can be seen in 4.2.1-2**.**

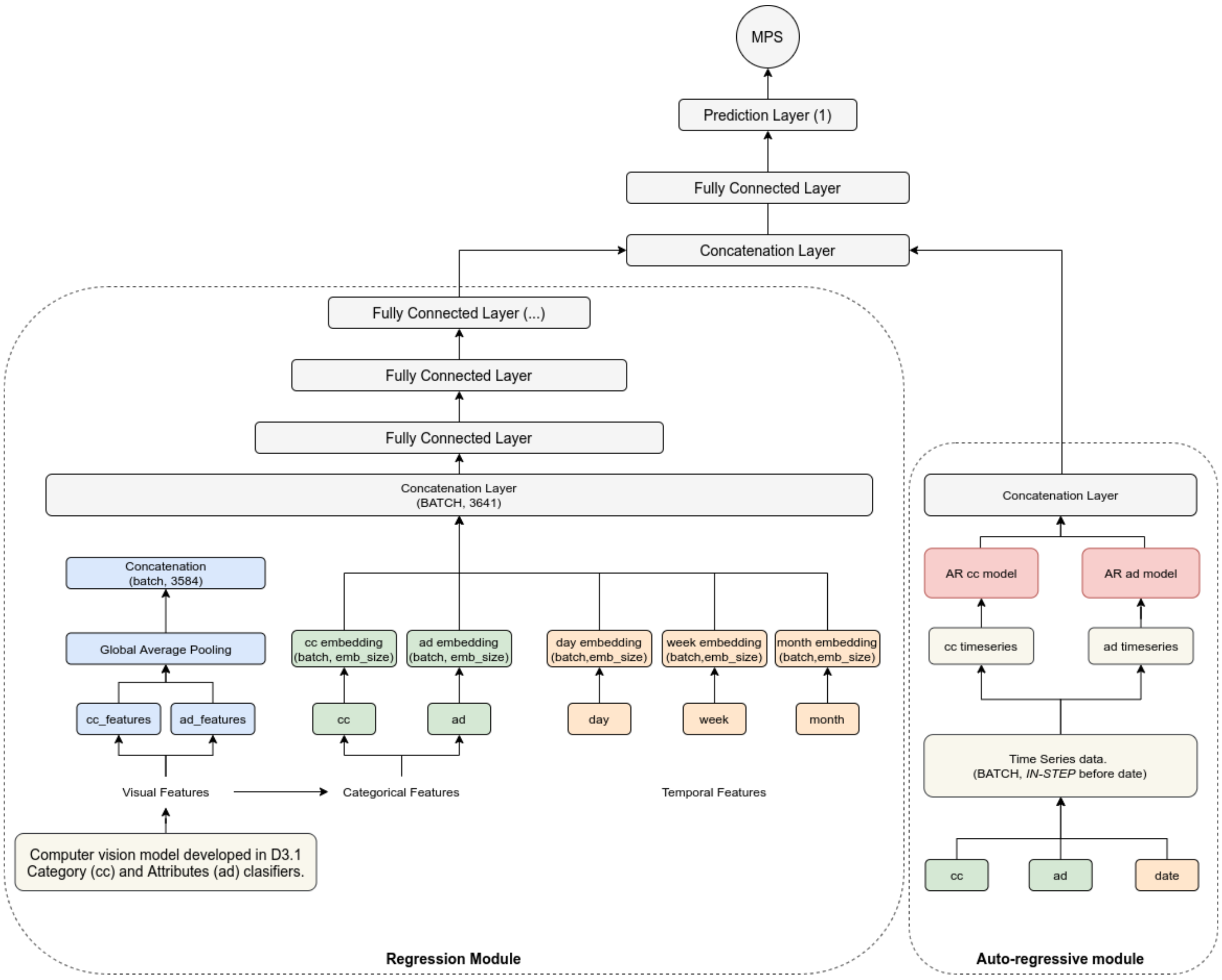

MPS
Prediction Layer (1)
Fully Connected Layer
Concatenation Layer
Fully Connected Layer (...)
Fully Connected Layer
Fully Connected Layer
Concatenation Layer (BATCH, 3641)
Concatenation (batch, 3584)
Global Average Pooling
cc_features
ad_features
cc embedding (batch, emb_size)
ad embedding (batch, emb_size)
day embedding (batch,emb_size)
week embedding (batch,emb_size)
month embedding (batch,emb_size)
cc
ad
day
week
month
Visual Features
Categorical Features
Temporal Features
Computer vision model developed in D3.1 Category (cc) and Attributes (ad) clasifiers.
Regression Module
Concatenation Layer
AR cc model
AR ad model
cc timeseries
ad timeseries
Time Series data. (BATCH, IN-STEP before date)
cc
ad
date
Auto-regressive module

Figure 4.2.1-2: Workflow of the *H-QAR* model combining the MLP Regressor (regression module) and two AR-NN models for a garment's fashion categories and attributes (auto-regressive module).

4.3 Demographics-based modelling

Both the *MLP Regressor* and the *H-QAR* architectures, as described in the previous section, are trained to predict the general popularity of a garment. They do not take into consideration any demographic information. Therefore, their predictions would be identical for all age groups and genders. For example, a newly designed "floral dress" would receive the same popularity score both for "females, age: 18-25" with "males, age:

40-50”. To overcome this limitation and perform more informed and segmented predictions, we adjusted both the *MLP Regressor* and the *H-QAR* to predict a garment’s popularity for different demographics groups. To this end, we added another embedding layer for the 26 combined demographic groups (age/gender) described in section 3.1.3. The resulting vector is concatenated with the categorical and visual features in the regression module and the network is re-trained on the *Mallzee market segmentation* dataset.

5. Empirical Results

5.1 Evaluation Protocol

In order to assess and fairly compare the performance of various models for each task (trend forecasting and garment popularity prediction) we had to define and follow precise evaluation protocols for each task regarding:

1. Data splits
2. Evaluation metrics
3. Multi-criteria selection
4. Grid-search of hyper-parameters

For each given task, we use the exact same data splits (training, validation and testing sets) in order to fairly compare the performance of different models. A detailed description of the data splits for each dataset was presented in Section 3.

All tasks (both trend forecasting and popularity prediction) can essentially be considered different forms of regression tasks. Therefore, we selected multiple evaluation metrics appropriate for regression. We mainly used the Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE) but also calculated the Pearson correlation and the Normalised Root Mean Squared Error (NRMSE) for the popularity prediction task. The NRMSE, uses the RMSE of the mean popularity scores for all instances in the testing set in order to normalise the RMSE rate of a given model. Therefore, NRMSE can be used as a comparison with the “mean prediction” baseline. It is worth mentioning that all metrics - with the exception of Pearson correlation - measure the error rate and divergence between the predicted and the ground truth values, thus the lower the metric’s value, the better a model has performed.

Having multiple evaluation metrics can be instrumental in identifying various strengths and weaknesses in a model. However, having multiple evaluation metrics make it more challenging to decide the overall best performance; since evaluation metrics can and often do come in conflict. In order to overcome this issue, we make use of TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) (Hwang & Yoon, 1981). TOPSIS is a multi-criteria decision analysis method which chooses the instance with the shortest geometric distance from the positive ideal solution and the longest geometric distance from the negative ideal solution. We defined the *MSE* and *MAE* as criteria for the trend forecasting task and the *MSE, MAE* and *Pearson correlation* for the popularity prediction task.

Finally, training machine learning models necessitates tuning the values of multiple hyper-parameters in order to identify the best performing combinations of each model. To this end, we performed extensive hyper-parameter grid-searches for every model in each task. We report the grid-searches that we performed in the following sections, alongside the empirical results of those experiments.

5.2 Fashion trends forecasting

For fashion trend forecasting we performed an extensive series of experiments on the Mallzee time series dataset and then retrained a selection of the best performing AR-NN models on the '*Paris to Berlin*' dataset in order to compare our models' performance against the current State-of-the-Art (SotA).

5.2.1 Mallzee time series dataset

A very significant parameter for trend forecasting models is the selection of the amount of *IN-STEPS* that the model will receive in order to predict the *OUT-STEPS* into the future. We could not decide beforehand which is the most appropriate combination of IN/OUT steps because this choice is conditional to the needs of fashion designers and retailers. Conditional to how much historical data is available and how further into the future they want to forecast. Therefore, we deemed it essential to experiment with various IN/OUT steps combinations for both daily and weekly aggregation and examine how different models behave under different settings. The detailed IN/OUT steps combinations that we selected to experiment with can be seen in Table 5.2.1-A. We were not able to experiment with larger IN/OUT steps combinations due to the limited size of the datasets. Setting IN:90 and OUT:30 days or having IN:12 and OUT:8 weeks did not leave enough samples to construct a testing set. Thus, we omitted those combinations from our experiments.

Table 5.2.1-A: Experimental grid-search of IN/OUT steps for daily or weekly time series aggregations.

Aggregation	IN-STEPS	OUT-STEPS
	7	1
	7	7
	30	1
	30	7
	30	30
Daily	60	1
	60	7
	60	30
	90	1
	90	7

	4	1
	4	4
	8	1
Weekly	8	4
	8	8
	12	1
	12	4

Alongside experimenting with daily and weekly aggregated time series, as well as various IN/OUT steps combinations, we selected a series of values of various hyper-parameters for each of the five AR-NN models, as seen in Table 5.2.1-B.

Table 5.2.1-B: Experimental grid-search for the five AR-NN models.

MODEL	HYPER-PARAMETER	VALUES
LSTM	# of LSTM Layers	1 2 3
	Units of LSTM Layers	512 512, 256 512, 256, 128
Conv	# of conv layers	1 2 3
	Convolutional filters	512 512, 256 512, 256, 128
	Kernel Size	3 7 14
	Stride	1 equal to kernel size
	Batch Normalization	True False
ConvLSTM	# of CONV layers	2 3
	Conv filters	512, 256 512, 256, 128
	# of LSTM layers	2 3
	LSTM Units	512, 256 512, 256, 128
	Kernel Size	3

		7 14
	Stride	1 equal to kernel size
	Batch Normalization	True False
Feedback LSTM	# of LSTM Layers	1 2 3
	Units of LSTM Layers	512 512, 256 512, 256, 128
	# dense layers	0 1 2
	Dense Layer Units	None 256 512
Transformer	# attention heads	2 4
	Embedding size	128 256 512
	# transformer blocks	1 2 3

Combining the IN/OUT step experiments (Table 5.2.1-B) for both category and attribute time series with the various hyper-parameters experiments (Table 5.2.1-C) resulting in a total of 3,194 experiments. All models are trained for a maximum of 100 epochs with the Mean Squared Error (MSE) loss function. We use an EarlyStop callback that monitors the loss function on the validation set with a patience of 5 epochs and returns the best performance epoch.

In the following tables, we report the best performing model (ranked based on TOPSIS) on each IN/OUT step combinations for either daily or weekly aggregation, trained for either the 22 categories or the 109 attributes time series. Alongside the AR-NN models we also ran a baseline approach, referred to as 'repeat last', which simply 'predicts' the next step by repeating the last time step.

Table 5.2.1-C: Results on daily time series for the 22 categories (ranked based on TOPSIS)

IN / OUT	Model	MAE	MAPE	MSE	RMSE	Hyper-Parameters
7 / 1	CNN	0.0898	13.1105	0.0137	0.1171	Batch norm : False Kernel size: 3 Conv filters: 512, 256, 128 Stride: 1
7 / 7	CNN	0.1022	14.32	0.0168	0.1299	Batch norm : False Kernel size: 3

						Conv filters: 512, 256, 128 Stride: 1
30 / 1	Feedback LSTM	0.0939	13.2343	0.0145	0.1203	LSTM layer units: 256 Dense layer units: 512
30 / 7	Transformer	0.1058	15.241	0.0183	0.1356	embedding size: 256 # attention heads: 2 # blocks: 1
30 / 30	CNN	0.1244	17.1785	0.0233	0.1525	Batch norm : False Kernel size: 14 Conv filters: 512, 256 Stride: 14
60 / 1	CNN	0.0965	12.414	0.0151	0.1228	Batch norm : False Kernel size: 14 Conv filters: 512, 256 Stride: 14
60 / 7	Transformer	0.102	13.278	0.0174	0.1317	embedding size: 512 # attention heads: 2 # blocks: 1
60 / 30	Transformer	0.1014	13.472	0.0176	0.1328	embedding size: 512 # attention heads: 4 # blocks: 3
90 / 1	Transformer	0.0876	10.4963	0.0135	0.1161	embedding size: 512 # attention heads: 4 # blocks: 2
90 / 7	Transformer	0.0918	11.046	0.0156	0.125	embedding size: 256 # attention heads: 2 # blocks: 3

Table 5.2.1-D: Results on daily time series for the 109 attributes

IN / OUT	Model	MAE	MAPE	MSE	RMSE	Hyper-Parameters
7 / 1	CNN	0.128	21.4643	0.0283	0.1684	Batch norm : False Kernel size: 3 Conv filters: 512, 256 Stride: 1
7 / 7	CNN	0.1323	23.0308	0.0302	0.1737	Batch norm : False Kernel size: 3 Conv filters: 512, 256, 128 Stride: 3
30 / 1	CNN	0.135	22.1342	0.0301	0.1734	Batch norm : False Kernel size: 7 Conv filters: 512, 256, 128 Stride: 7
30 / 7	CNN	0.1441	23.4007	0.0331	0.1818	Batch norm : False Kernel size: 14 Conv filters: 256 Stride: 1

30 / 30	CNN	0.1545	25.5786	0.0373	0.1932	Batch norm : False Kernel size: 14 Conv filters: 512, 256 Stride: 1
60 / 1	Repeat Last	0.1089	16.4635	0.0312	0.1766	-
60 / 7	Transformer	0.1433	21.1738	0.0346	0.1859	Embedding size: 128 # attention heads: 2 # blocks: 1
60 / 30	ConvLSTM	0.155	21.9808	0.0364	0.1907	Batch norm : True Kernel size: 14 Conv filters: 512, 256 LSTM units: 512, 256 Stride: 14
90 / 1	Repeat Last	0.0744	10.0689	0.0211	0.1452	-
90 / 7	Repeat Last	0.0967	12.6444	0.0293	0.1712	-

Table 5.2.1-E: Results on weekly time series for the 22 categories

IN / OUT	Model	MAE	MAPE	MSE	RMSE	Hyper-Parameters
4 / 1	Repeat last	0.0695	9.0481	0.0079	0.0894	-
4 / 4	Conv	0.0922	12.9243	0.0131	0.1145	Batch norm : False Kernel size: 2 Conv filters: 256 Stride: 1
8 / 1	Repeat last	0.0721	8.6033	0.0083	0.0912	-
8 / 4	ConvLSTM	0.0849	9.8841	0.0102	0.1012	Batch norm : False Kernel size: 2 Conv filters: 512, 256, 128 LSTM units: 512, 256 Stride: 2
8 / 8	Transformer	0.1267	14.5205	0.0249	0.1577	Embedding size: 128 # attention heads: 2 # blocks: 1
12 / 1	Repeat Last	0.0574	6.1621	0.0057	0.0754	-
12 / 4	Transformer	0.0928	10.0216	0.0124	0.1112	Embedding size: 512 # attention heads: 4 # blocks: 1

Table 5.2.1-F: Results on weekly time series for the 109 attributes

IN / OUT	Model	MAE	MAPE	MSE	RMSE	Hyper-Parameters
4 / 1	Repeat last	0.086	11.3858	0.0128	0.1132	-
4 / 4	Conv	0.1086	14.8323	0.0177	0.1331	Batch norm : False Kernel size: 2 Conv filters: 512, 256 Stride: 2
8 / 1	Repeat last	0.0836	10.3025	0.0125	0.1119	-
8 / 4	Conv	0.1191	14.4441	0.0202	0.1421	Batch norm : False Kernel size: 2 Conv filters: 512, 256 Stride: 2
8 / 8	Transformer	0.1704	20.2937	0.0439	0.2095	Embedding size: 256 # attention heads: 2 # blocks: 1
12 / 1	Repeat Last	0.0698	8.1236	0.0104	0.1019	-
12 / 4	Repeat Last	0.1041	11.6099	0.0196	0.1399	-

We notice that CNNs tend to perform better for lower IN/OUT step values, while the Transformer tends to perform better with higher IN/OUT step values. However, the Repeat Last baseline appears numerous times in Tables 5.2.1-E and 5.2.1-F, which present results on the weekly aggregated time series. This is mostly observed in the cases where the OUT-step is equal to one. One possible explanation is that while most time series from the Mallzee time series dataset show a gradual but continuous upward trend, there is an abrupt increase during April-May of 2020 (Figure 3.1.4-1). After that point, most time series plateau for consecutive steps. Those dates are only present in the validation (03-06/2020) and testing sets (06-10/2020); since the time series are sorted ascendingly and split accordingly. Similar trends - of abrupt increases and consecutive plateaus - were not present in the training set and thus could not have been modelled by the AR-NNs. Thus, simply repeating the previous step (Repeat Last baseline) yields a better performance in those cases. In contrast, in the cases where the OUT step is higher than one, AR-NNs tend to perform better for the weekly aggregated time series. This shows their effectiveness and their superiority in contrast to the repeated prediction.

In order to assess whether AR-NN models tend to perform better on daily or weekly data, and on which IN/OUT-step combinations, we retrieved the top-50 best performances for category time series and top-50 attribute time series (based on TOPSIS, excluding the repeat last baseline). This process was necessary for deciding which models and IN/OUT-steps to use while building the H-QAR model. As can be seen in Table 5.2.1-G, the majority of AR-NN models perform better when using the weekly aggregated data instead of the daily data, for both category and attributes time series. The implications are discussed in section 5.3.1 during designing the series of experiments around the H-QAR model.

Table 5.2.1-G: Comparing overall performance in daily and weekly aggregated time series.

Input time series	IN-STEPS	Model counts	Total model count per input
Daily attributes	90	6	8
	7	2	
Weekly attributes	4	30	**42**
	8	10	
	12	2	
Daily categories	7	9	22
	90	8	
	60	3	
	30	2	
Weekly categories	4	15	**28**
	12	11	
	8	2	

5.2.1 'Paris to Berlin' time series dataset

Following Al-Halah & Grauman (2020), we trained multi-target AR models on all 46 attributes for 50 cities, which translates into 2200 time series and experimented with 1 through 8 time lags. We performed extensive grid search for five AR-NN models: LSTM, CNN, ConvLSTM, Feedback LSTM and Transformers. The grid search was similar to that in the Mallzee time series dataset. We experimented with different numbers of layers and number of units/filters/transformer blocks but we did not experiment with adding batch normalisation layers nor altering kernel sizes in the CNN-based models. Rather we kept the best values - as found in the Mallzee dataset - for those hyper-parameters. We performed a total of 481 experiments on the *Paris to Berlin* dataset.

A Feedback LSTM model with a single LSTM layer of 256 units and a dense layer of 512 units was able to achieve the best performance on the Paris to Berlin dataset. In Table 5.2.1-H we report the best performance of each of our AR-NN and compare them with the models reported in the original Paris to Berlin paper. We can see that our Feedback-LSTM is able to surpass the Influence & Coherence model proposed by Al-Halah & Grauman (2020) by decreasing -3.43% the MAE and -4.08% the MAPE metric.

Table 5.2.1-H: Comparison between our AR-NN models and various models reported in the 'Paris to Berlin' paper.

	Model	MAE	MAPE
~~Naive~~	Gaussian	0.1301	33.23

	Seasonal	0.0925	22.64
	Mean	0.0908	23.57
	Last	0.0893	22.20
	Drift	0.0956	23.65
Per city	Fashion Forward (Al-Halah et al., 2017)	0.0779	19.76
	AR	0.0846	21.88
	ARIMA	0.0919	23.70
	GeoModel (Mall et al., 2019)	0.0715	17.86
All cities	VAR	0.0771	19.25
	Paris to Berlin (Influence & Coherence)	0.0699	17.38
	OURS (Feedback LSTM)	**0.0675**	**16.67**
	OURS (Conv)	0.0693	17.17
	OURS (LSTM)	0.0705	17.59
	OURS (ConvLSTM)	0.0846	21.48
	OURS (Transformer)	0.0854	21.37

5.3 Garment popularity prediction

For the task of garment popularity prediction we performed an extensive series of experiments on the *Mallzee MPS* dataset using both the MLP Regressor and the H-QAR architecture. Thereafter, we re-created a selection of experiments on the *SHIFT15m* dataset in order to validate our findings and compare ourselves with the current SotA.

5.3.1 Mallzee MPS dataset

In *garment popularity prediction* we essentially attempt to predict the popularity of a garment on a specific date, based on a series of features. In the *Mallzee MPS* dataset the following features were available: the garment's colors, the price of the product, the time we wanted to predict (day, week, month), the visual features as well as the fashion labels (category and attributes) of the garment (extracted from the computer vision models developed in D3.1).

In our initial experiments we wanted to assess the usefulness of different feature combinations. To this end, we created a series of experiments using all possible features combinations. We found that, using auxiliary features (price and colors) alongside visual and category/attribute labels, could improve the model's performance. The MLP Regressor using all available features resulted in 0.2391 MAE and 0.0801 MSE, while only using visual features and fashion labels had a slightly worse performance, with 0.2416 MAE and 0.0815 MSE and solely using visual features resulted in 0.2432 MAE and 0.0821 MSE. However, we decided to exclude 'colors' and 'price' and keep only the visual features alongside the fashion labels because in the deployed version models these features may be missing, especially in the Designer app.

Following the experiments on the different features combinations, we performed a grid-search experiment in order to tune various hyper-parameters of the *MLP Regressor*. The hyper-parameters and their values can be seen in Table 5.3.1-A (resulting in 162 experiments in total) while the top-5 performances (based on TOPSIS) can be seen in Table 5.3.1-B.

Table 5.3.1-A: Grid-search experiment for the MLP Regressor.

Hyper-parameter	**Values**
Embedding size	8, 16, 32
Loss function	MSE, MAE, LogCosh
Number of layers	2, 3, 4, 5, 6, 7
Dense units (1st fully-connected layer)	2048, 1024
Dense units (Additional fully-connected layers)	512 1024 512, 256 1024, 512 1024, 512, 256 1024, 512, 256, 128 1024, 512, 256, 128, 64 1024, 1024, 512, 256, 128, 64 1024, 1024, 1024, 512, 256, 128

Table 5.3.1-B: Top-5 hyper-parameter combinations (based on TOPSIS) of the MLP Regressor, trained on the Mallzee MPS dataset. Bold denotes the best performance per metric.

Parameters	MAE	MAPE	MSE	RMSE	Pearson	NRMSE
Loss function: ‘MSE’, Embedding size: 8, Dense Units: [2048, 1024, 512, 256]	0.2389	98.8565	**0.0805**	**0.2837**	**0.433**	**0.9024**
Loss function: 'MSE', Embedding size: 32, Dense Units: [1024, 1024, 512, 256]	0.2392	100.032	0.0809	0.2843	0.4301	0.9044
Loss function: 'LogCosh', Embedding size: 32, Dense Units: [2048, 1024, 512, 256, 128]	**0.2384**	**98.0751**	0.0811	0.2848	0.4312	0.9057
Loss function: 'MSE', Embedding size: 8, Dense Units: [1024, 1024, 512, 256]	0.2395	101.596	0.0808	0.2843	0.4297	0.9042
Loss function: 'MSE', Embedding size: 16,	**0.2384**	99.8409	0.0811	0.2848	0.4304	0.9059

Dense Units: [2048, 1024, 1024, 1024, 512, 256, 128]

Proceeding with the *H-QAR* model on the *Mallzee MPS* dataset, we used TOPSIS in order to assess the best performing models for both garment popularity prediction and fashion trend detection and then combine them. Firstly, the analysis on garment popularity prediction showed that the top-5 best performing models exhibited very small divergences. Therefore, we decided to select only the top-2 highest performing hyper-parameter combinations and use them for the *H-QAR* model. Those were, two *MLP Regressors* with:

- 4 fully-connected layers, having 2048, 1024, 512, 256 units respectively, embedding size of 8 dimensions and was trained with the MSE loss function,
- 5 fully-connected layers, having 2048, 1024, 512, 256, 128 units each, embedding size of 32 dimensions and was trained with the LogCosh loss function

We did not experiment with more *MLP Regressor* architectures because it would simply increase the number of total experiments, a time-consuming and resource-intensive process. Secondly, using TOPSIS on the fashion trend detection experiments exhibited slightly more complicated results due to having more parameters; including the number of IN-STEPS and OUT-STEPS, having 'daily' or 'weekly' aggregated time series, having 5 different AR-NN architectures alongside multiple hyper-parameter for each. As we saw in Table 5.2.1-F, more models performed better on average for weekly aggregated data for both category and attributes time series. More specifically, 60% of the top-50 AR-NN models trained on the attribute time series performed best while having an input of 4 IN-STEPS, followed by 20% with 8 IN-STEPS. For category time series, 30% of the models performed best on average when having 4 IN-STEPS as input, followed by 22% for 12 IN-STEPS. Overall, for both weekly attributes and categories, 45% of the top-100 models performed better while having 4 IN-STEPS. However, we decided to select 12 IN-STEPS for categories and 8 IN-STEPS for attributes that also contained the 4 IN-STEPS. Thereafter, we selected the best performing hyper-parameter combination for each of the 5 *AR-NN* (CNN, LSTM, ConvLSTM, Feedback LSTM and Transformers) on weekly aggregated category and attribute time series respectively. We created a series of experiments combining the two best performing *MLP Regressor* models and the best five *AR-NN* models for categories and attributes respectively. This resulted into a total of 2 * 5 * 5 = 50 experiments. The top-5 performances (based on TOPSIS) can be seen in Table 5.3.1-C. All H-QAR experiments in the top-5 performed better with the following *MLP Regressor* architecture: 5 fully-connected layers with 2048, 1024, 512, 256, 128 dense units, and embedding size of 32 and 'LogCosh' as the loss function. Therefore, we only report only the category and attribute AR-NN models in Table 5.3.1-C.

Table 5.3.1-C: Top-5 hyper-parameter combinations (based on TOPSIS) of the *H-QAR* model; trained on the Mallzee MPS dataset. Bold denotes the best performance per metric.

Category AR-NN	Attributes AR-NN	MAE	MAPE	MSE	RMSE	Pearson	NRMSE
CNN	Feedback LSTM	**0.2299**	**95.66**	**0.0762**	**0.2761**	**0.4801**	**0.8781**

CNN	ConvLSTM	**0.2299**	99.38	0.0768	0.2771	0.4759	0.8814
ConvLSTM	ConvLSTM	0.2311	100.88	0.0769	0.2774	0.4758	0.8821
CNN	Transformer	0.2305	98.27	0.0768	0.2772	0.4746	0.8816
Transformer	Feedback LSTM	0.2307	96.31	0.0769	0.2773	0.4738	0.8819

Comparing the performance of the MLP Regressor with the H-QAR, shown in Table 5.3.1-D, we can observe that our hypothesis has been validated. H-QAR, by utilizing quasi auto-regression, is able to improve upon all 7 evaluation metrics. We also report the performance of a baseline Linear Regression model for comparison.

Table 5.3.1-D: Comparing the performance of 1) Baseline Linear Regression, 2) MLP Regressor and the 3) H-QAR on the Mallzee MPS dataset.

Model	**MAE**	**MAPE**	**MSE**	**RMSE**	**Pearson**	**NRMSE**
Linear Regression	0.2601	114.33	0.0905	0.3009	0.288	0.9574
MLP Regressor	0.2389	98.85	0.0805	0.2837	0.433	0.9024
H-QAR	**0.2299**	**95.66**	**0.0762**	**0.2761**	**0.4801**	**0.8781**

5.3.2 SHIFT15m dataset

As mentioned in Section 3.2.2, the *SHIFT15m* dataset is not intended to be used for certain predefined tasks, instead it is multi-purpose and can be utilized for multiple tasks defined by the user. Among other tasks, the creators of *SHIFT15m* used the dataset for popularity prediction of fashion outfits. While *outfit popularity prediction* is not our target task, we trained our *MLP regressor* to ensure the compatibility of our model. We used the exact same settings[3] as the authors of SHIFT15m and re-trained all their models on the same split of the dataset. As can be seen in Table 5.3.2-A, our MLP Regressor is able to outperform the baseline models.

Table 5.3.2-A: Comparing our MLP Regressor with the models used by Kimura et al. (2021)

Model	**MAE**
LinearRegression	9.329
TheilSenRegressor	9.863
HuberRegressor	29.73
RANSACRegressor	10.316
DecisionTreeRegressor	12.851
MLP Regressor (ours)	**9.242**

However, our central objective was to use the SHIFT15m dataset for evaluating our models' performances (for MLP Regressor and H-QAR) on a dataset other than the

[3] github.com/st-tech/zozo-shift15m/tree/main/benchmarks#regression-for-the-number-of-likes

Mallzee MPS dataset. Since garment-level popularity prediction is not available in SHIFT15m we re-mapped the dataset from the original outfit-level to the garment-level. This process is described with details in Section 3.2.2.

We follow the exact same workflow for SHIFT15m as with the *Mallzee MPS* dataset. We trained 1) the MLP regressor and 2) H-QAR model from scratch on the *SHIFT15m* dataset. Informed by our experiments on the *Mallzee MPS* dataset we limited the grid search of the various hyper-parameters. We selected the best performing hyper-parameter combinations for each of the 5 AR-NN models and the top-4 best performing *MLP Regresso*r models. We trained the 4 MLP regressors on their own and in combinations with the 5 different AR-NN models. This translated into 24 experiments in total for each of SHIFT-2016-2020 and SHIFT-2017-2020 (48 experiments in total).

Table 5.3.2-B reports the best performances (based on TOPSIS) for a baseline Linear Regression model, the MLP regressor and the H-QAR; for both versions of the *SHIFT15m* dataset. As with the *Mallzee MPS* dataset, our hypothesis that "quasi auto-regression can improve the performance of garment popularity prediction models" is again validated; since in both versions of *SHIFT15m* H-QAR outperforms the MLP Regressor as well as the baseline Linear Regression model.

The overall best performance was achieved, using the 2016 - 2020 SHIFT dataset, by the H-QAR model, combining a MLP Regressor of 4 fully-connected layers with 1024, 512, 256, 128 dense units and an embedding size of 4 and a LSTM AR-NN with 1 LSTM layer of 512 units. The network was trained with the MSE loss function for 15 epochs with a learning of 0.0001.

We can also observe that all three models performed better on the larger version of SHIFT15m (2016 - 2020) which means that they could benefit from a larger dataset. Unfortunately it was not feasible to utilize a larger set of the dataset due to limitations in computational resources.

Table 5.3.2-B: Comparing the performance of 1) Baseline Linear Regression, 2) MLP Regressor and the 3) H-QAR on the SHIFT15m dataset.

Dataset	Model	MAE	MSE	RMSE	Pearson	NRMSE
SHIFT 2017 - 2020	Linear Regression	0.1744	0.0445	0.2110	0.2739	0.9618
	MLP Regressor	0.1615	0.0396	0.1991	0.4267	0.9074
	H-QAR	0.1599	0.0386	0.1966	**0.4536**	**0.8964**
SHIFT 2016 - 2020	Linear Regression	0.1495	0.0323	0.1798	0.2751	0.9616
	MLP Regressor	0.1418	0.0297	0.1725	0.3947	0.9224
	H-QAR	**0.1403**	**0.0287**	**0.1694**	0.4262	0.9060

5.4 Market segmentation models

In order to carry out informed predictions for different demographic groups, we retrained each of the best performing models (baseline linear regression, MLP Regressor, H-QAR) on the Mallzee market segmentation dataset which included popularity scores per age group and gender. In Table 5.4-A we report the highest performance for each of the three models. Both MLP Regressor and H-QAR consisted of 4 fully-connected layers (2048, 1024, 512, 256 units), an embedding layer of 32 dimensions and was trained with the LogCosh loss function. The highest performing H-QAR also utilized CNN and ConvLSTM AR-NN models for fashion categories and attributes respectively. We can again observe H-QAR improving upon the MLP Regressor and that the overall performance of both models is better in the *Mallzee market segmentation* dataset than their counterparts in the *Mallzee MPS* dataset.

Table 5.4-A: Comparing the performance of 1) Baseline Linear Regression, 2) MLP Regressor and the 3) H-QAR on the *Mallzee market segmentation* dataset which included popularity scores per age group and gender.

Model	MAE	MSE	RMSE	Pearson	NRMSE
Linear Regression	0.2733	0.1031	0.3212	0.3899	0.9213
MLP Regressor	0.2224	0.0803	0.2833	0.5874	0.8126
H-QAR	**0.2178**	**0.0785**	**0.2802**	**0.6024**	**0.8036**

5.5 Limitations

We have seen the ability of H-QAR to surpass the baseline linear regression as well as the MLP Regressor on both Mallzee and SHIFT15m datasets. On average, the H-QAR performed 2.9% better in terms of MAE and 3.8% in terms of MSE; on Mallzee's MPS and market segmentation datasets. However, H-QAR, on top of the visual features, requires dense time series data and continuous monitoring in order to function properly in real time. Collecting, filtering and monitoring the time series of fashion categories and attributes is a considerably resource-intensive process. Therefore, it is arguable if the observed advantage in performance justifies the use of H-QAR over the MLP Regressor in all cases. In cases where dense time series are not already available, we consider the MLP Regressor to be an adequate and cost-effective alternative.

Moreover, as mentioned in Section 3.1.4, we have identified two irregularities in the time series of Mallzee data: the lack of seasonal variation and an upward trend for all fashion categories during the later months of 2020. We plan on further examining and addressing these irregularities in order to further improve the overall performance of the popularity prediction model.

6. Implementation and API integration

6.1 Implementation overview

To serve the eTryOn applications with the models developed in deliverables D3.1 and D3.2, an API was developed to expose methods for invoking the models and retrieving results from them. As part of this deliverable an API has been deployed on Amazon Web Services (AWS) using two core services, namely API Gateway and Sagemaker. The former (API Gateway), provides a way to connect access to the outside world with our ML models, control user access, and check and format data coming in and out of the service. The latter (Sagemaker), is a solution provided by AWS to help facilitate the deployment and maintenance of machine learning models. The API is accessed by posting HTTPS requests to endpoints on the following domain https://etryon.mallzee.com. Currently there is one exposed model allowing access to classifications produced in D3.1 and work is underway to expose the models produced in D3.2.

The developed API will consist of two modules, the *computer vision* and the *garment popularity prediction* modules. A designer will be able to upload an image of a new garment design. The designer will also have to select a future date and the gender and age group. The image will be sent to the *computer vision* module to extract the visual features (from both the category and attributes classifiers) and predict the categories and attributes of the depicted garments (D3.1). The visual features, fashion labels, date and the target demographic group are forwarded to the *garment popularity prediction* module which will calculate the popularity of the new garment design for the requested demographic group and date (D3.2). A visualised example can be seen in Figure 6.2.

6.2 Example request to exposed API

We present an example request to our API along with the corresponding responses.

POST https://etryon.mallzee.technology/attributes

INPUT DATA: { “img_url”:

“https://mlz-data-849976359079-public.s3.eu-west-1.amazonaws.com/2540f523-e194-58ed-95da-a75d68dd2232.jpg”,

“date”: “1-11-2021”,

“demographic_group”: “18-25 & female”

}

RESPONSE from the *Computer Vision API* module:

[

{'odject_type': 'fullbody',

```
        'category': 'dresses',
        'attributes': ['floral', 'tea'],
        'category_classifier_features': array([ -9.41425  ,  -9.32711  , ...,  -13.67753  ,
        -1.0519407]),
        'attributes_classifier_features': array([-15.135585, -22.501026, ..., -10.55093 ,
        -15.449522]),
        'box': array([0.24207637, 0.11850899, 0.9276604 , 0.9024589 ])
       }
]
```

RESPONSE from the *Garment Popularity Prediction* API module:
[{‘popularity_score’: 63.21}]

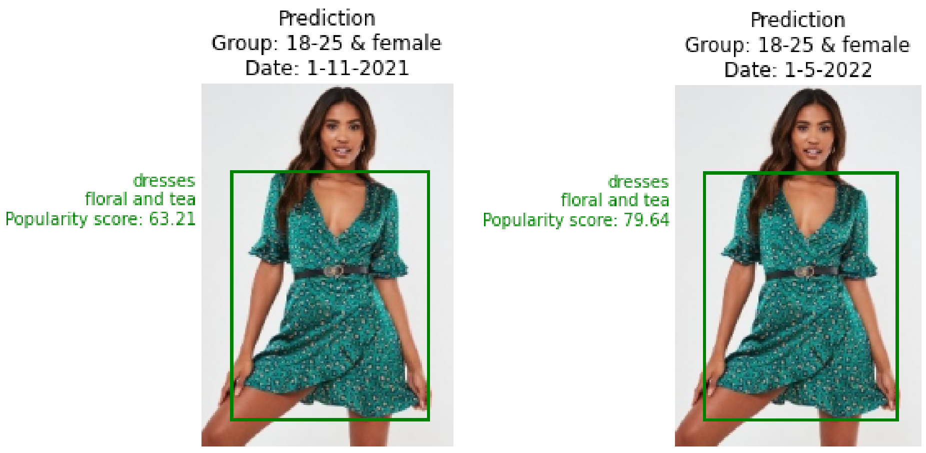


Figure 6.2: Inference example on the same garment for “females 18-25” on two different dates.

7. Conclusions

Our objective, as outlined in deliverable D3.2, was to identify trends in fashion and develop machine learning models capable of predicting the popularity of new garment designs on different market segments, defined by gender and age groups. We developed a hybrid approach that utilizes the visual features of garments (using computer vision models from D3.1) and historical time series of the garment's category and attributes. We named this model H-QAR, because it combines a multi-layered perceptron for analysing a garment's visual features and a quasi auto-regressive neural network for modelling the time series of the garment's category and attributes.

In order to train and evaluate the machine learning models on the defined tasks, we had to create a large-scale image fashion dataset with historical popularity scores. We used the raw data from Mallzee's databases, cleaned them and developed a pipeline, which downloaded the images and employed the computer vision models from D3.1 for extracting the garment's category, attributes and visual features. We defined the mallzee performance score (MPS) as the target variable; the metric of popularity.

Thereafter, we performed an extensive series of experiments in order to tune the hyper-parameters of the various machine learning models and yield the best possible performance from them; thus ensuring the validity of our findings. Moreover, we trained the various models on both Mallzee and public datasets in order to compare our methods with the current state of the art and examine their reproducibility and usefulness. More specifically, we used the *Paris to Berlin* dataset for the task of forecasting trends on fashion attributes (Al-Halah & Grauman, 2020). Our Feedback LSTM method was able to achieve MAE: 0.0675 and MAPE: 16.67, thus surpassing the previous best performing method which was MAE: 0.0699 and MAPE:17.38.

Furthermore, we used the *SHIFT15m* dataset for the task of popularity prediction. The dataset is multi-objective; meaning that it can be used for numerous tasks. Kimura et al., (2021) initially used *SHIFT15m* for outfit-level popularity prediction. On this task, our MLP Regressor scored MAE: 9.242 surpassing the models used in the SHIFT15m paper, where the highest score was MAE: 9.329. However, our main interest in this deliverable was garment-level popularity prediction. Therefore, we remapped the SHIFT15m from the outfit to the garment-level and used the visual features and the garment category time series to re-train our MLP Regressor and the H-QAR model. Empirical experimentation on both Mallzee's datasets and the SHIFT15m dataset showed the ability of H-QAR to surpass both the baseline linear regression and the MLP Regressor.

Our plans for the immediate future is to extract visual features (from D3.1), fashion trends and popularity scores (from D3.2) and utilize them - alongside other features - in the development of recommendation services that will be used in the eTryOn applications.

8. References

Al-Halah, Z., & Grauman, K. (2020). From Paris to Berlin: Discovering fashion style influences around the world. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 10136-10145).

Al-Halah, Z., Stiefelhagen, R., & Grauman, K. (2017). Fashion forward: Forecasting visual style in fashion. In Proceedings of the IEEE international conference on computer vision (pp. 388-397).

Borovykh, A., Bohte, S., & Oosterlee, C. W. (2017). Conditional time series forecasting with convolutional neural networks. arXiv preprint arXiv:1703.04691.

Ghader, H. (2021). An empirical analysis of phrase-based and neural machine translation. arXiv preprint arXiv:2103.03108.

Graves, A. (2013). Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Hwang, C. L., & Yoon, K. (1981). Methods for multiple attribute decision making. In Multiple attribute decision making (pp. 58-191). Springer, Berlin, Heidelberg.

Kimura, M., Nakamura, T., & Saito, Y. (2021). SHIFT15M: Multiobjective Large-Scale Fashion Dataset with Distributional Shifts. arXiv preprint arXiv:2108.12992.

Lim, B., & Zohren, S. (2021). Time-series forecasting with deep learning: a survey. Philosophical Transactions of the Royal Society A, 379(2194), 20200209.

Lo, L., Liu, C. L., Lin, R. A., Wu, B., Shuai, H. H., & Cheng, W. H. (2019, September). Dressing for attention: Outfit based fashion popularity prediction. In 2019 IEEE International Conference on Image Processing (ICIP) (pp. 3222-3226). IEEE.

Ma, Y., Ding, Y., Yang, X., Liao, L., Wong, W. K., & Chua, T. S. (2020, June). Knowledge enhanced neural fashion trend forecasting. In Proceedings of the 2020 International Conference on Multimedia Retrieval (pp. 82-90).

Mall, U., Matzen, K., Hariharan, B., Snavely, N., & Bala, K. (2019). Geostyle: Discovering fashion trends and events. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 411-420).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

Zhao, B., Lu, H., Chen, S., Liu, J., & Wu, D. (2017). Convolutional neural networks for time series classification. Journal of Systems Engineering and Electronics, 28(1), 162-169.