2AMM10 Assignment 1: Learning fashion features

Tuesday 22nd April, 2025

Introduction

Pradah Hinson is a visionary fashion designer who has built a reputation for creating stylish and trendy clothing. Inspired by the growing trend of online shopping, she has decided to modernize her business by integrating AI into her fashion house. Pradah wants a smart system that can understand fashion items and organize them efficiently, making online shopping effortless for her customers. However, traditional approaches rely heavily on predefined categories, limiting their ability to adapt to new trends. Instead of simply assigning labels, she needs an AI that can learn meaningful representations of fashion items—capturing similarities, differences, and emerging trends.

As AI experts, you have been recruited to develop this intelligent system. Your challenge is to design models that can extract and learn useful features from fashion images, making it possible to not only categorize known styles but also recognize and adapt to unseen ones. The project consists of multiple tasks, gradually increasing in complexity to build a robust solution. Are you ready to shape the future of fashion AI? Let's get started!

The dataset

The world of fashion is full of spies! Pradah is anxious about potential data leaks and decided that the proof of concept that you will develop for her future AI solution will be done on a public dataset. Her data engineers have curated the following dataset: *Fashion Product Images*. The full (curated) dataset contains around 44k images and each image is associated with one of the 69 possible article types (classes). One important aspect of the project is the presence of "new classes" which are classes not seen during training. In order to evaluate the solution with the presence of new classes, the data engineers have done the following split of the dataset:

- *Main classes train*: train dataset with the first 39 classes (size: 36k)
- *Main classes test*: test dataset with the first 39 classes (size: 2k)
- *New classes test*: test dataset with the last 30 classes (size: 1.7k)

Figure 1: Examples of images from the *Fashion Product Images* dataset

- *Main classes support*: support dataset[1] with the first 39 classes (size: 2k)
- *New classes support*: support dataset with the last 30 classes (size: 1.7k)

Task 1: Help Pradah organize her collection

Designers often say that before crafting the more advanced fashion items, you should first master the basics. In this spirit, Pradah would like to start the project with a classical classification task, i.e. training a neural network with a classification loss such as cross entropy. Given the picture of a fashion item, the model has to assign the correct article type (class). For this task you have to:

1. Design an appropriate neural network architecture for classifying the fashion images
2. Train your model on *main classes train* with cross entropy loss
3. Assess the performance of the trained model on *main classes test*. As not all classes are equally represented in the test set[2], you are required to report the performance in term of accuracy and balanced accuracy. A good model is expected to reach an accuracy and balanced accuracy of at least 75%.

Let D be the dataset on which we want to evaluate the performance of the model $\hat{f}$. The dataset contains C classes and can be decomposed in $D = D_1 \cup ... \cup D_C$ where $\forall c \in [1, C] : D_c = \{(x, y) \in D | y = c\}$. We can then define accuracy and balanced accuracy as:

$$\text{accuracy} = \frac{1}{|D|} \sum_{(x,y)\in D} \mathbf{1}\left(\hat{f}(x) = y\right) \tag{1}$$

$$\text{balanced accuracy} = \frac{1}{C} \sum_{c=1}^{C} \frac{1}{|D_c|} \sum_{(x,c)\in D_c} \mathbf{1}\left(\hat{f}(x) = c\right) \tag{2}$$

where $\mathbf{1}(a = b)$ is the indicator function which equals 1 if $a = b$ and 0 otherwise.

Task 2: How to deal with new fashion trends?

Fashion is constantly evolving, new article types emerge each year while others slowly disappear. It is not realistic for the fashion house to constantly rebuild and retrain a model to adapt to the new types of articles. Moreover, classifying articles is not the final goal of the AI models we would like to build. We want to learn dense representations of the images (feature vectors) that can be used to assess the similarity between two articles. These dense representations will be used later for downstream tasks (see task 3 and 4).

For this task, Pradah wants you to build a model that is able to encode images of fashion items into feature vectors that can be used to assess their similarities. More precisely, given the image of a fashion article, she wants images of the same article type to be closer to it in the feature space than images of different types. You are required to do the following:

1. Train a model with the same architecture as in task 1[3] but this time with a loss appropriate for the requirements given above. Think also about an appropriate strategy to select samples during the training process
2. At this stage, we still want to assess the model performance in a classification setting, however we also consider the case where some classes have not been used during training. In order to classify an image from a given test set, you have access to a support set that contains images and their classes. How can you use the support set to classify test images (without retraining the model)?

[1] The notion of support set is introduced in task 2

[2] The class imbalance is the same between all main classes datasets. Similarly the class imbalance is the same between all new classes datasets

[3] Use the 39 output neurons (without softmax) as the values of the feature vector

3. Assess the performance of your classification method in the following scenarios:

Scenario	Support set	Test set	Description
1	Main classes train	Main classes test	Evaluating performance on main classes using training data as reference
2	Main classes support	Main classes test	Evaluating performance on main classes using a separate support set
3	New classes support	New classes test	Evaluating performance on new classes
4	Main classes support + New classes support	Main classes test + New classes test	Evaluating performance on mixed main and new classes

In terms of performance, your model might not achieve extremely high accuracy in scenario 3 and 4 given that the model needs to generalize to unseen classes at test time. As a ballpark estimate, you can expect to achieve an accuracy and balanced accuracy of at least 45% in scenario 3.

Task 3: Embracing recommendation systems

After successfully building a model that can handle both existing and new fashion trends, Pradah wants to enhance her customers' shopping experience with personalized recommendations. She envisions a system where customers viewing a specific fashion item will be shown similar items they might be interested in purchasing. While browsing Pradah's online store, customers expect to see relevant recommendations. However, fashion recommendation is a delicate matter—irrelevant suggestions can frustrate customers and potentially drive them away. As Pradah puts it:

"I'd rather show no recommendation than suggest something completely unrelated to what my customer is looking for."

For this task, you will leverage the feature representation model you developed in task 2 to build an intelligent recommendation system with the following properties:

1. Given an input image, the system should retrieve the 3 most similar items from the available catalog (represented by your support set). The 3 most similar items constitute a recommendation.
2. A recommendation is considered correct if at least one of the three recommended items belongs to the same class as the input image
3. The system should be selective—it should only display a recommendation when it has sufficient confidence that it will be correct

How can you build such a system?

For this task, you are required to do the following:

1. Using your model from task 2, design a recommendation system that provides recommendations along with a confidence measure for each recommendation. We want our system to assess confidence without relying on the classes present in the support set (catalog). The goal is to develop a system that can also be used with an unlabeled catalog.
2. The decision to show a recommendation or not to the user depends on whether the associated confidence score is above a threshold. The choice of this threshold is influenced by the percentage of incorrect recommendations among the recommendation shown to the user that we tolerate (error rate) as well as the percentage of recommendations that we would like to be shown to the user (coverage). Evaluate how these two quantities (error rate and coverage)

evolve with the threshold by creating a plot with "error rate (%)" on the x-axis and "coverage (%)" on the y-axis. For this plot, use the combined datasets from scenario 4 (task 2), where the support set represents your product catalog and the test set represents customer inputs. What do you observe?

Task 4: Beyond labels

Pradah believes that fashion is not just about categories—it's about understanding the nuances that make each piece unique. In this task, you will delve into the feature space learned in task 2.

Step 1: Visualizing the feature landscape

1. Use a dimensionality reduction method to project the learned features of the first 10 classes onto a 2D space.
2. Generate scatter plots of the learned features from the 10 first classes for both the *main classes train* and *main classes test* datasets.
3. Examine the plots: Identify which classes form distinct clusters and which ones overlap. Which classes appear easy to distinguish and which do not ?

Step 2: Verifying our assumption with a confusion matrix

1. Create a confusion matrix using:
   - **Support set:** the first 10 classes[4] from the *main classes support* dataset.
   - **Test set:** the first 10 classes from the *main classes test* dataset.
2. Assess whether the classes that overlapped in your scatter plots also show higher misclassifications rates in the confusion matrix.

Step 3: A closer look at shirts vs T-shirts

Shirts and T-shirts are often hard to tell apart. To understand this better:

1. Compute the centroids (average feature vectors) for the shirt and T-shirt classes in both train and test data.
2. For each centroid, display the closest image (in the feature space), representing a "typical" shirt and T-shirt.
3. By selecting samples close to the line connecting the two centroids, visualize the transition between shirt and T-shirt in the feature space.

Step 4: Representation of higher level classes

In our analysis of the first 10 classes, we observed that some classes exhibit greater similarity than others. For example, the feature embeddings of a shirt and a T-shirt are often closer to each other than a shirt and sunglasses. This suggests a hierarchy in our feature space. More generally, we expect items belonging to different article types but sharing the same higher-level category (e.g., casual shoes and sport shoes are both shoes) to remain relatively close to each other in the feature space. To test this hypothesis, we will leverage the hierarchical structure in our dataset. Our dataset includes a coarser classification level called *categories*. The 39 *article types* that we've used as main classes until now can be merged into 20 broader *categories*. Using the datasets of the scenario 2, evaluate the performance of the model[5] if the task is to classify *categories*.[6]

[4] The integer associated to the class is between 0 and 9

[5] Do not retrain the model, use the model trained in task 2

[6] Details on how to switch from *article types* to *categories* are provided in Jupyter notebook skeleton

Deliverables

The submission of this assignment is done in ANS. There are two deliverables:

- Implementation of the solution
- Description and explanation of your approach, formatted as answers to the questions in ANS.

For the implementation, a skeleton Jupyter Notebook with functions that can load the provided data for each task are provided. The notebooks also contain short bits of self-explanatory code on how the data is formatted. Please submit your code in the provided skeleton Jupyter notebook. You need to upload the Jupyter Notebook code (.ipynb file extension) and a PDF version (.pdf file extension) with the execution output. The maximum upload size is 25MB. So, you may need to clean some image output from the code Notebook before uploading. If you are using Google Colab, you can generate the PDF by going to "File → Print → Save as PDF". Please generate the PDF after running all cells of your solution (with possibly images removed if necessary for file size constraints).
Please note that you should train all models from scratch for the assignments in this course. The use of pre-trained models is not allowed.