Fashion Image Classification

Udacity Learning Engineer Nanodegree Capstone Proposal

Jahyun Koo
November 6th, 2019

Project Overview

This document is a proposal for the Capstone Project of the Udacity Machine Learning Engineer Nanodegree Program.
The goal of the capstone project is to create a Deep Learning Model that categorized fashion product images in [the Fashion Product Images Dataset]

Domain Background

Personal motivation is the main reason for selecting this project.

I'm a developer creating a fashion SNS and I have to solve a lot of problems similar to this proposal
I need to learn about image processing and deep learning, as well as develop the ability to apply it in practice.

This is the biggest reason.

In the service i'm developing, when a use uploads a fashion image, the operator need to tag it.

For example, this image is tagged as hood.

We are not using an automated classification algorithm. Therefore, the operator must tag it manually.
This is a very inefficient task. It takes a lot of time and effort.

We have collected approximately 50,000 tagged images to automate this task.
If the algorithm can automate this task, it will save a lot of time and money.

I'm the person in charge of classifying images in a team.
So i need to learn and experience algorithms like deep learning for image classification.

I tried to find a dataset as close as possible to what I was trying to do.

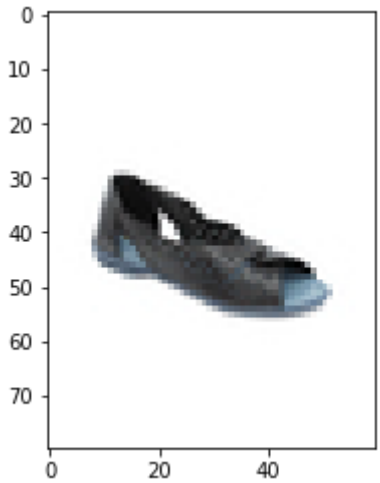
When i looked up the dataset in Kaggle, there was a dataset very similar to the problem I had to solve.
I want to apply an image classification algorithm to this dataset.

Problem Statement

This is a multi-class classification problem. Inputs are image of fashion product, and the goal is to predict category label.
Given an image of a fashion product, the algorithm will identify and estimate of the image's category.

Datasets and Inputs

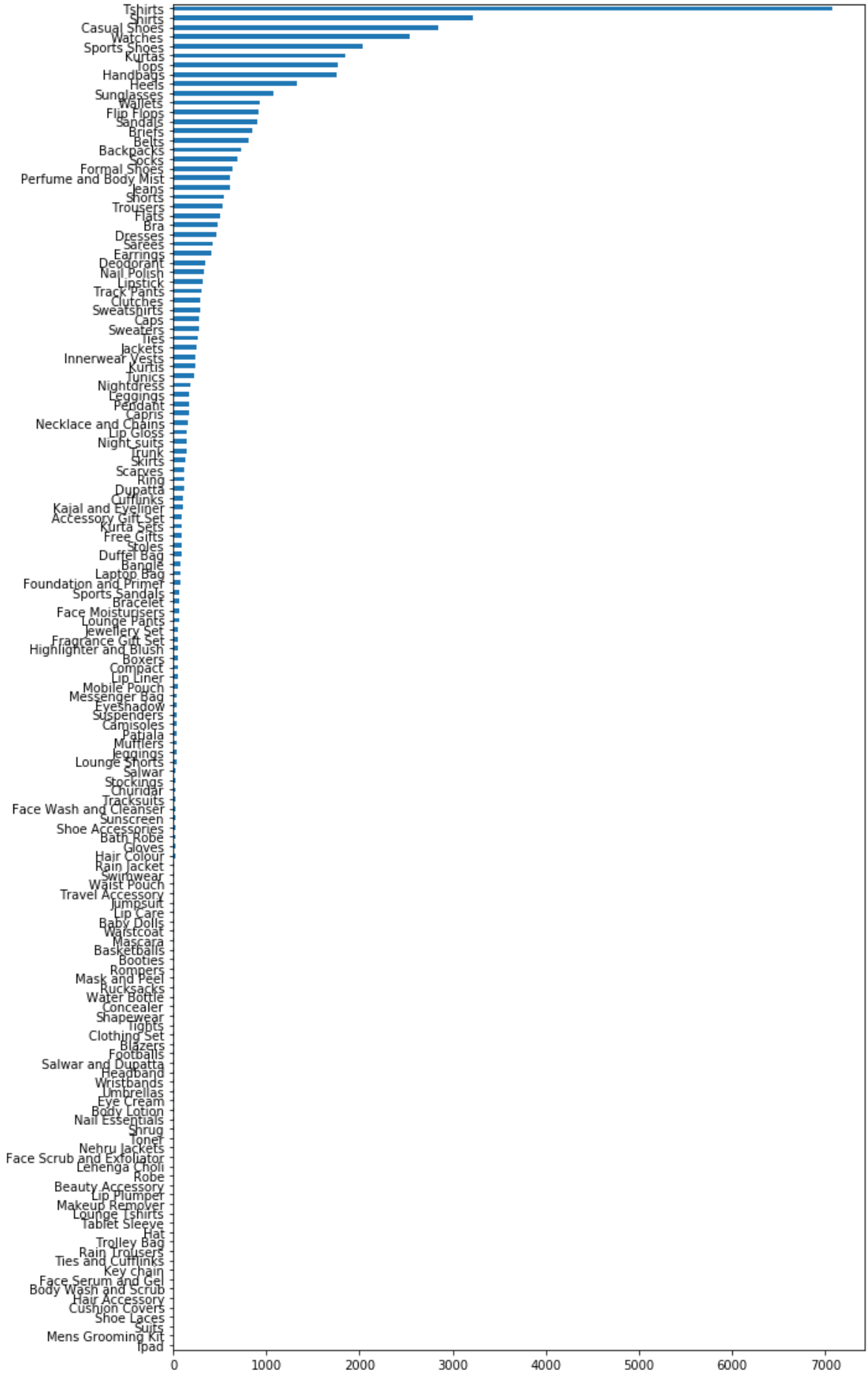
Each image has information such as `master category`, `sub category`, `article type`, `color`, `season` for each image.
Images's resolution is 80 x 80 x 3. For example, below is an image of the `Heels` category.



(Kaggle has a high resolution image of the same dataset, but because of the lack of resources to train the model, I use an image dataset with such a low resolution.)

I want to predict `article type` information from several types. This is because the information is most similar to what the actual user wants.

Below is the distribution of categories.



The biggest problem with looking at the distribution is that the number of images in a particular category is very small. Like this, if there is not enough image of a certain category, learning of that category may not work well. Therefore, we will also consider the handling of this situation.

Solution Statement

This problem is clearly a supervised learning problem because the image(fashion product) is labeled.
Deep learning is known as the best solution for image classification problems like this. So I will also apply deep learning to this problem.

First, i will use pandas and numpy to gain some understanding of the data.
And, to build a baseline model, I will create and train a very simple deep learning CNN model.

This model is very simple and can be trained quickly.
If this model gets good accuracy, we will work to tune the model.

Otherwise, simple models may not be able to learn or may take too long time.
In this situation, it is known to use ***transfer learning***. Transfer learning is a method of reusing a model or knowledge for another related task. Transfer Learning has the following advantages.

- Improved baseline performance
- Model-development time
- Improved final performance

Most importantly, we can get good performance in a short development period. When applying deep learning to real industry rather than research, it's important to create a model as soon as possible. First of all, it is important to release the service on time.

Therefore, it is not practical to train all models from scratch. In addition, research has shown that using Transfer Learning improves model robustness and uncertainty estimates.

My goal is to apply deep learning to a live commerce service, so I will use this model because I think it is most appropriate to use Transfer Learning.

Benchmark Model

We plan to create a simple CNN model, train it, and use it as a benchmark model. This model is a very simple model with two CNN layers. Since the resolution of the image is not large, I decided that deep learning model with small layers would be a good baseline.

```
class SimpleCNNModel(torch.nn.Module):
    def __init__(self):
        super(SimpleCNNModel, self).__init__()
        # L1 ImgIn shape=(?, 80, 60, 3)
        #    Conv     -> (?, 80, 60, 32)
        #    Pool     -> (?, 40, 30, 32)
        self.layer1 = torch.nn.Sequential(
            torch.nn.Conv2d(3, 32, kernel_size=3, stride=1, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(kernel_size=2, stride=2))

        # L2 ImgIn shape=(?, 40, 30, 32)
        #    Conv      ->(?, 40, 30, 64)
        #    Pool      ->(?, 20, 15, 64)
        self.layer2 = torch.nn.Sequential(
            torch.nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(kernel_size=2, stride=2))

        # Final FC 20x15x64 inputs -> 142 style category
        self.fc = torch.nn.Linear(20*15*64, 142, bias=True)
        torch.nn.init.xavier_uniform_(self.fc.weight)
```

```
    def forward(self, x):
        out = self.layer1(x)
        out = self.layer2(out)
        out = out.view(out.size(0), -1)
        out = self.fc(out)
        return out
```

Evaluation Metrics

The simplest way is to use accuracy. However, as shown in the dataset exploration above, the distribution of categories is biased, so accuracy-based assessments do not work. If one category has more than the other, performance can be misunderstood for accuracy. Therefore, measuring with accuracy alone is never a good way.

To accurately evaluate model, we need a way to measure model's performance using the Confusion Matrix. Confusion matrix is a table that is often used to describe the performance of a classification model.

	True condition positive	**True condition negative**
Predicted condition positive	TP = True positive	FP = False positive
Predicted condition negative	FN = False negative	TN = True negative

Several evaluation metrics can be derived from the above Confusion Matrix.

- Accuracy = (TP + TN) / (TP + FP + TN + FN)
- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)

We can calculate F1 score from the metrics given above. F1 score is the harmonic mean of precision and recall. The score is multiplied by 2, so if both precision and recall are 1, the score is 1.

- F1 = (2 * Precision * Recall) / (Precision + Recall)

F1 score is more useful than accuracy, especially if dataset has imbalanced category distribution.

Project Design

- Programming Languages and Libraries
    - python 3
    - jupyter notebook
    - pytorch, numpy, matplotlib, ...
- Data Exploration and Visualization
    - view image
    - basic statistics and understanding of the dataset
    - made some visualization like the quality of images, the distribution of category
- Data Preprocessing
    - perform basic cleaning and processing if needed
    - split the data into train, validaiton, test sets
    - feature scaling
- Implement benchmark model

    - implement simple CNN model
    - training and testing
- Implement transfer learning model
    - implement transfer learning CNN model
    - training and testing
- Model tuning
    - Fine tune the model's hyperparameter
- Testing
- Write report

References

- Learning to Learn: Knowledge Consolidation and Transfer in Inductive Systems
- A Survey on Deep Transfer Learning
- Comprehensive Survey on Transfer Learning