DATS6303 DEEP-LEARNING

DEBORAH AINA

PROFESSOR DR. AMIR JAFARI

CLASSIFYING CLOTHING CATEGORIES USING COMPUTER VISION

Introduction and Motivation

The global fashion industry is a colossal market, valued at approximately $1.7 trillion in 2023. Within this vibrant ecosystem, the United States stands out with a market size of $343.70 billion. On a per capita basis, Americans lead globally, spending an average of $1460 annually on clothing and footwear. This high level of expenditure is indicative of the country's strong consumer culture, particularly among Gen Z, where 36% report purchasing new clothing at least once every month.

Despite the significant advancements in computer vision across various sectors such as healthcare, sports, and automotive, its application in the fashion industry remains relatively under explored.

Our project, "Fashion Finder", aims to bridge this gap by utilizing machine learning models to classify images of people donned in various outfits. These models are designed to recognize and categorize different clothing items and attributes such as t-shirts, pants, dresses, glasses and other four six categories in various poses. By doing so, we aspire to provide a tool that not only enhances consumer experience but also offers valuable insights to retailers and designers by understanding current trends and customer preferences more profoundly. This initiative is not just about classifying clothing but in future transforming how we
perceive and interact with fashion using artificial intelligence and create bespoke personalized apparels for consumers.

The project was divided into parts that enabled each teammate to focus on their strengths. I spearheaded the proposal, presentation, my own model and worked together with my teammate on the final group report. My teammate worked on his chosen models, app and together on the presentation and final group report,

Data Preprocessing

This dataset was originally designed for a challenge focused on creating bounding boundaries of automatic product detection in fashion images.

The dataset comprised approximately 50,000 images of people wearing a variety of clothing types in a variety of poses. This dataset also came with a Json file of the categories or attributes of the images. I started by preprocessing the data. This process was done together with my teammate over approximately four hours zoom session including other different virtual sessions.

Each image is recorded in the data showing the different attributes. The dataset was grouped by the columns Image Id and Class Ids and each Class Id combined by the underscore delimiter. A new column called Category name is added to the Data Frame to reflect the categories or attributes. A column called Split is also created to separate the data into training and validation set using 80-20 split. A custom data class is used to load the samples one at a time, transform the images and the labels ready to be processed into one hot encoded label. The bulk of the work done was in preprocessing the data, addressing class imbalance and finding the ideal hyperparameters to use in training the model since we used pretrained models.

A very important victory was finally getting the class weight calculation to work. I had designed to use this in Exam2 but did not get the chance to do it. This significantly improved the model's performance.

To calculate class weights, we first converted the categorical labels into a binary matrix where each row represents an image, and each column represents a class. This matrix was achieved using the Pandas get dummies function on the Category column of our training data:
label matrix = xdf_dtrain["Category"].str.get_dummies(",")
Next, we computed the frequency of each class across all samples:
class frequencies = label matrix.sum()
total samples = class frequencies.sum()
Using these frequencies, we determined the weight for each class by dividing the total number of samples by the product of the frequency of the class and the number of classes, ensuring that under-represented classes are given higher importance during model training:
class weights = total samples / (class frequencies * len(class frequencies))

To apply these weights during the training phase, we calculated sample weights for each image in the dataset. As it is Multilabel, we were doing the sum of the weights of each class present as the weight of the image. But it was taking a lot of time, so we got the same operation done faster multiplying the binary label matrix with the class weight matrix.

InceptionNet

InceptionNet is a convolutional neural network (CNN) architecture that Google developed to improve upon the performance of previous CNNs on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). The reason for choosing this pretrained model is because it uses inception modules which is a combination of m * m sized kernels specifically 1 * 1, 3 * 3 and 5 * 5. The result is that these smaller and varying kernel sizes will learn a combination of local and global features from the input data (images). These feature maps created at different scales are then concatenated together to form a better representation of the input data and this is known as the inception module.
Applying batch normalization and other regularization techniques through layers of convolving and pooling, the pretrained model returns a SoftMax classification however, the task is at hand is to predict a multilabel classification data, the last layer of the pretrained model is transformed to a linear layer and trained. The Inception Net model expects the input data(image) to come in as 299 * 299 so the image had to be resized to match this size. Using pytorch torchvision transforms version 2 method, we were able to apply both geometric and functional transformations to the input data(images)

Training and Experiments

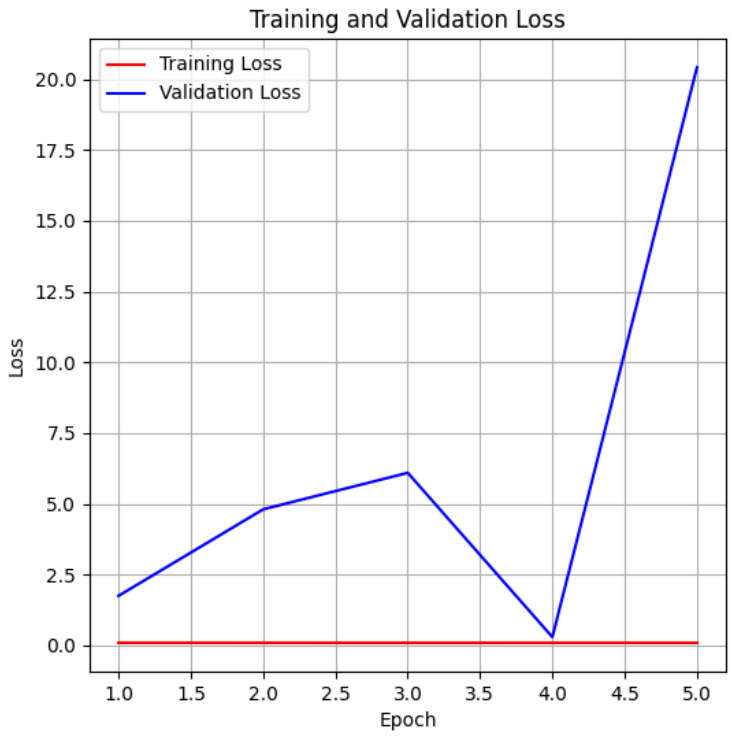
The model was trained using different batch sizes and epochs as well as learning rates to determine the ideal hyperparameters to use. Experimenting with different epochs showed different performances. The figures Figure 1 and Figure 2 below show the validation loss increasing, decreasing and increasing sharply after that. Another experiment was using the pretrained model alone versus in addition to the pretrained model, training the model on the last layer using the Fashion dataset. Doing the latter improved the model's performance on the validation set. A custom loss function is used in training the model.

FocalLoss function is designed to address class imbalance by down weighting the easy examples or labels such that the contribution of these easy examples to the overall loss value is small. The traditional Cross entropy function applies equal weights to each class i.e. for a given predicted probability p, the loss value calculated will be the same for any class. To solve this problem, if the predicted probability of a class is low, we penalize the loss heavily and if the predicted probability is high, we do not penalize the loss. We introduce two parameters, alpha and gamma. Gamma is the modulating factor, if we increase gamma, it changes the loss function curve and extends our criteria of well-classified examples, consequently extending the range of probabilities where the loss function is low. Gamma reduces the loss contribution from easy examples. Alpha on the other hand, is the weighting factor, (we see this in the class weight calculation). is the inverse class frequency. alpha at t is alpha for positive class and 1 - alpha for negative class. This helped our model focus on harder examples during training.

Figure 1 FocalLoss Equation

$$FL(p) = \begin{cases} -\alpha(1-p)^{\gamma} log(p), & y = 1 \\ -(1-\alpha)p^{\gamma} log(1-p), & otherwise \end{cases}$$

Figure 2 Training and validation loss trends for Inception Net for epoch 4



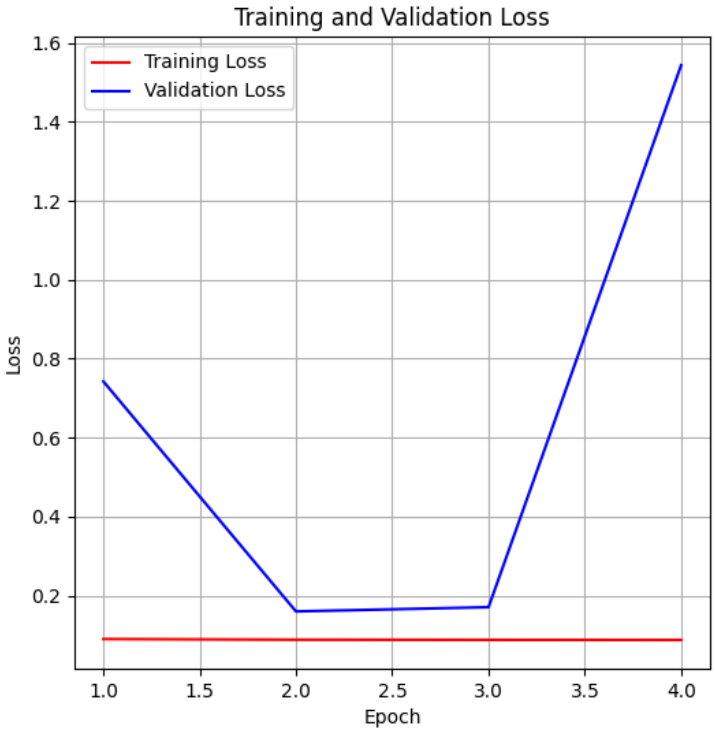


Figure 3 Training and validation loss trends for Inception Net for epoch 4

Metrics

To evaluate the effectiveness of our models on the validation set, we analyzed their performance using a variety of metrics, including Accuracy, F1 Score, Precision, and Recall. These metrics provide insights into how well each model predicts the correct clothing categories.

Conclusion

Our project successfully tackled the complex task of multi-label classification of fashion apparel using state-of-the-art machine learning models such as CNNs, ResNets, InceptionNet and transformers. Among these, my teammates model the Vision Transformer (ViT) model stood out demonstrating superior accuracy and generalization capabilities across various clothing categories.

Furthermore, the integration of these models into a Streamlit-based application provided a seamless and user-friendly platform for real-time fashion classification. This application allows users to effortlessly upload images and receive immediate, reliable classification results, making it an excellent tool for both fashion enthusiasts and industry professional.

I am also working on combining the power of the pre-trained RetinaNet using Resnet50 as its backbone with attribute features for better classification. This will combine the image feature extraction and attributes feature extraction concatenated together in the last layer with a log sigmoid function. The code should be ready and uploaded to GitHub for your consideration.

References

1. Official Pytorch documentation
2. Pytorch Multi Label Classifier Github example
3. Enhancing Multi-Class Classification with Focal Loss in PyTorch
4. Focal Loss Explained Focal Loss Explained | Papers With Code