FASHION ATTRIBUTES CLASSIFICATION CHALLENGES

AI6126 Project 1

Riemer van der Vliet
G2304212K

2024 - March - 29

1 Introduction

This report outlines the methodology and findings of the Fashion Attributes Classification Challenge, an individual endeavor focused on leveraging machine learning to categorize fashion images by attribute. Employing a dataset of 6,000 images, this project distinguishes itself through the use of Vision Transformers (ViT). The objective is to accurately predict multiple attribute labels for each image, a task that demands a model with nuanced understanding of the dataset. Particularly as this dataset is heavily unbalanced.

1.1 The dataset

Fashion Attributes Classification Challenge dataset comprises 6,000 images, thoughtfully partitioned into sets of 5,000 for training and 1,000 for validation. This rich dataset is annotated with 26 distinct attribute labels spread across six primary categories, framing a complex multi-label classification scenario. Each image is tagged with six attributes, one from each category. Some examples can be seen in figures 1.

Figure 1: First 4 entries of the dataset.

The dataset is regularized with a simple transformer as can be seen in the code in The code and in code block 2. Where a RandomVerticalFlip is added to increase the number of datapoints, but no RandomHorizontalFlip is added as this is not representative of the test set.

```
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
    transforms.RandomVerticalFlip(p=0.5)
])
```

Figure 2: Code snippet for dataset transformation

1.1.1 Weight regularization

Addressing the challenge of dataset imbalance, a weight regularization strategy has been implemented. You can see the relative weights as they are computed in figure 3. This approach aims to normalize the representation of classes within the model, mitigating the skewness caused by varied class frequencies. As such when calculating the loss function the weights are used to balance the loss function.

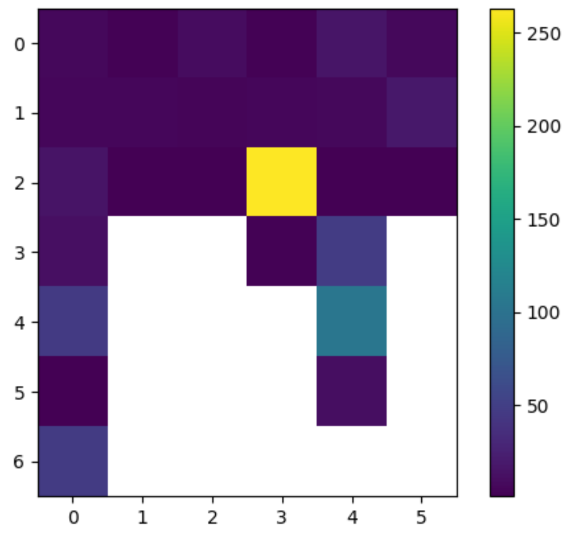

Figure 3: Weight distribution of classes

1.1.2 Loss function

Because of the weight imbalance as mentioned earlier and the multi-label classification task, a weighted binary cross-entropy loss function has been employed. This function is designed to assign different weights to each class based on their frequency, ensuring that the model learns to predict all classes effectively. This approach is crucial for optimizing the model's performance and enhancing its ability to classify fashion attributes accurately.

1.2 Description of model

In training the model I have taken particular note in making sure the model was able to capture subclasses within the dataset as such we will employ a Multitaskhead with some shared Relu layers with dropout and some task specific layers

1.2.1 ViT

The Vision Transformer (ViT) model marks a paradigm shift from traditional convolution neural networks to a transformer-based approach that treats image patches as sequences for classification tasks. This model capitalizes on the transformer's ability to capture long-range dependencies which might work well in a dataset with big differences within image projection. The precise model used is a vit-base-patch16-224-in21k from Hugging Face. Notably the author did not read the documentation of this model and accidentally used a model trained on 21k images opposed to the required 10k.

1.2.2 MultiTaskHead

At the heart of our model lies the MultiTaskHead module, engineered to concurrently predict multiple attribute labels. This functionality is vital for the Fashion Attributes Classification Challenge, ensuring the model's efficiency and efficacy in multi-label classification. The module's architecture, integrated within a pre-trained ViT framework, demonstrates a robust approach to handling the intricacies of fashion attribute classification. The printed model can be seen in the appendix 10.

1.2.3 Model Architecture

The model employs a pre-trained Vision Transformer (ViTForImageClassification) as its backbone, featuring an intricate network of embeddings, encoders, and a MultiTaskHead for classification. The encoder, composed of 12 transformer layers, processes sequences of image patches through a series of self-attention mechanisms and feed forward networks. This is followed by a MultiTaskHead classifier, which includes shared layers for general feature processing and task-specific layers designed to output predictions for each attribute category. The structure and parameterization of this model underscore its potential to capture and interpret the complex patterns inherent in fashion imagery, providing a detailed and nuanced understanding of fashion attributes.

2 Results

figure 7 shows the training loss during training, while figure 5 illustrates the model training with a high learning rate. This exploration methods indicated that a more optimal solution would be somewhere in the middle. As can be seen in the training loss graph in figure 7 the training is smooth up to this point but seems to stagnate. The full training sets can be found in 1. Where I have employed crossing to quickly pinpoint optimal parameters.

From this I conclude that my loss function is not able to accuracy project the loss. This is also visible in the validation loss graph in figure 6. The validation loss is not decreasing as expected. This is a clear indication that the model is not learning as expected.

Layers Unfrozen	**Epochs**	**Learning Rate**	**Dropout Rate**	**Average Class Accuracy**	**Total Accu**
1	10	0.0031	0.22	0.23	0.52
1	20	0.00031	0.22	0.58	0.75
1	20	0.00031	0.3	0.56	0.72
1	20	0.0004	0.22	0.54	0.72
1	50	0.000031	0.22	0.54	0.6

Table 1: Model Training Parameters: Layers Unfrozen, Epochs, Learning Rate, Dropout Rate, Average Class Accuracy, Total Accuracy. Batch Size: 32

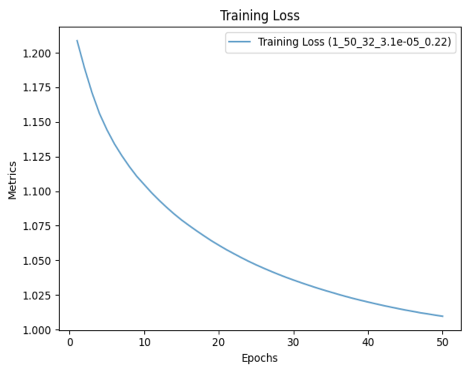

Figure 4: Here the model is trained over 50 epochs with low learning rate. The image shows the loss during training.

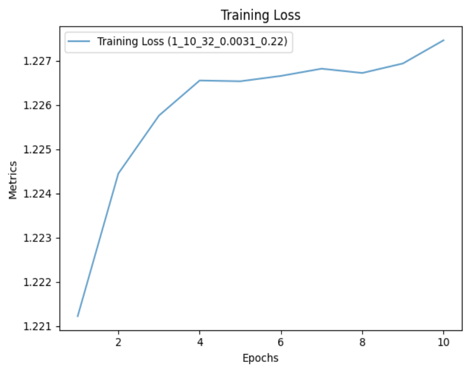

Figure 5: Here the model is trained over 10 epochs with high learning rate.

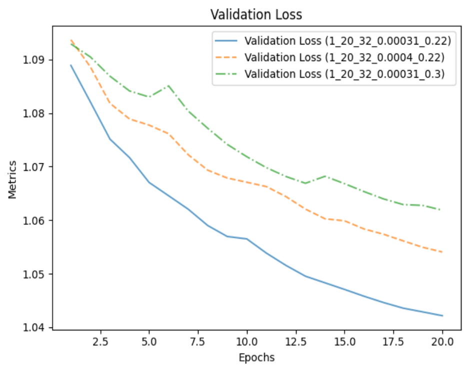

Figure 6: Validation Loss during Training

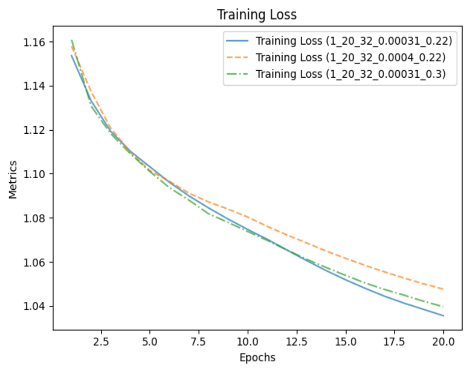

Figure 7: Training Loss during Training

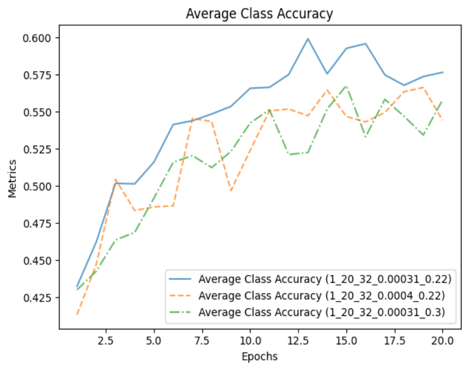

Figure 8: Average Accuracy during Training

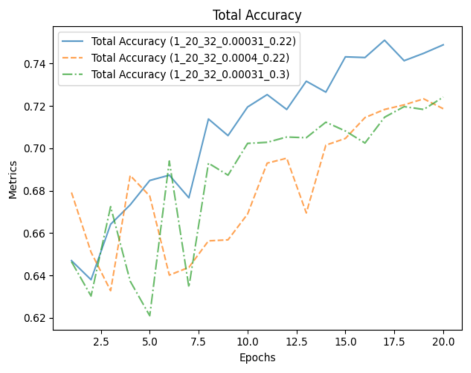

Figure 9: Total Accuracy during Training

3 Discussion

Based on the training and validation loss graphs shown in Figures4 and5, it is evident that the model's performance is not satisfactory. The training loss seems to stagnate, indicating that the model is not learning as expected. Similarly, the validation loss does not decrease as expected, further confirming that the model is not performing optimally. It is clear that the chosen ViT (Vision Transformer) model is not able to effectively capture the complex patterns inherent in fashion imagery. This might due to using a model trained on 21k images instead of the required 10k images, which might not have good capabilities to represent a smaller dataset.

Despite the suboptimal performance, the model achieved the best results with the parameters: 1 layer unfrozen, 20 epochs, batch size of 32, learning rate of 0.00031, and dropout rate of 0.22. These parameters were selected based on a combination of experimentation and cross-validation to find a balance between model performance and computational efficiency. However, it is important to note that these parameters still did not yield satisfactory results.

Further investigation is required to determine the optimal parameters for the model. However, it is worth noting that training a ViT model requires substantial computational resources, which are currently unavailable to me. This limitation hinders the ability to conduct extensive testing and parameter tuning as well as retrain the model using a 10k variant.

Despite the suboptimal performance, the model achieved a final test score of 0.57721. The model was unable to express the data better and trials with optimizers such as Optuna did not yield better results (not shown in report).

In conclusion, the current model's performance falls short of expectations. The limitations of the chosen ViT model and the lack of computational resources hindered the ability to achieve optimal results. Further research and experimentation are necessary to overcome these challenges and improve the model's performance.

4 Bibliography

5 Appendix

Listing 1: model architecture

```
MultiTaskPretrained(
    (pretrained): ViTForImageClassification(
        (vit): ViTModel(
            (embeddings): ViTEmbeddings(
                (patch_embeddings): ViTPatchEmbeddings(
                    (projection): Conv2d(3, 768, kernel_size=(16, 16), stride=(16
                )
                (dropout): Dropout(p=0.0, inplace=False)
            )
            (encoder): ViTEncoder(
                (layer): ModuleList(
                    (0-11): 12 x ViTLayer(
                        (attention): ViTAttention(
                            (attention): ViTSelfAttention(
                                (query): Linear(in_features=768, out_features=768
                                (key): Linear(in_features=768, out_features=768,
                                (value): Linear(in_features=768, out_features=768
                                (dropout): Dropout(p=0.0, inplace=False)
                            )
                            (output): ViTSelfOutput(
                                (dense): Linear(in_features=768, out_features=768
                                (dropout): Dropout(p=0.0, inplace=False)
                            )
                        )
                        (intermediate): ViTIntermediate(
                            (dense): Linear(in_features=768, out_features=3072, b
                            (intermediate_act_fn): GELUActivation()
                        )
                        (output): ViTOutput(
                            (dense): Linear(in_features=3072, out_features=768, b
                            (dropout): Dropout(p=0.0, inplace=False)
                        )
                        (layernorm_before): LayerNorm((768,), eps=1e-12, elementw
                        (layernorm_after): LayerNorm((768,), eps=1e-12, elementwi
                    )
                )
            )
            (layernorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        )
        (classifier): MultiTaskHead(
            (shared_layers): Sequential(
                (0): Linear(in_features=768, out_features=1024, bias=True)
                (1): Dropout(p=0.2, inplace=False)
                (2): ReLU()
                (3): Linear(in_features=1024, out_features=1024, bias=True)
                (4): Dropout(p=0.2, inplace=False)
                (5): ReLU()
                (6): Linear(in_features=1024, out_features=1024, bias=True)
```

no external sources have been used that need to be cited.