FAE-Net: Fashion Attribute Editing via Disentangled Latent Conditioning in Diffusion Models - Supplementary Material

1. Background

1.1. DDPM, DDIM and LDM

: Denoising diffusion probabilistic models (DDPMs) [3] learn to transform noise into images by reversing a predefined forward noising process. Unlike GANs, they have stable training and natural ways to incorporate conditioning c using either class labels or text. During the forward noising process, a clean image x_0 is corrupted with a Markov chain of incremental gaussian noise, which can be written as: $q(x_t \mid x_{t-1}) = \mathcal{N}\big(x_t; \sqrt{1-\beta_t}\, x_{t-1},\, \beta_t I\big)\,, \quad t = 1 \dots T$, where β_t is a small fixed noise schedule. After some notational changes and reparameterization, the equation to sample x_t directly from x_0 can be written as: $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon$, $\epsilon \sim \mathcal{N}(0, I)$ where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. As t increases in x_t, we eventually reach isotropic gaussian noise. During the reverse process (sampling), a neural network like U-Net is learned to reverse the noising, conditioned on c. The network is trained using the denoising objective:

$$\mathcal{L} = \mathbb{E}_{x_0, c, t, \epsilon} \left[\|\epsilon - \epsilon_\theta(x_t, t, c)\|^2 \right]$$

where ϵ_θ predicts the noise term ϵ in the reparameterization above.

DDPM sampling requires many steps since it is stochastic and follows a Markov chain. To overcome this, *Song et al.* [6] proposed denoising diffusion implicit models (DDIM), a family of non-Markovian deterministic and stochastic samplers that share the same training objective as DDPM but allow deterministic trajectories, which are useful for inversion and faster sampling. Given the model's predicted noise ϵ_θ and a user-chosen noise level σ_t, the DDIM reverse update is defined as:

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\, \epsilon_\theta(x_t, t, c)}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2}\, \epsilon_\theta(x_t, t, c) + \sigma_t z$$

where $z \sim \mathcal{N}(0, I)$, σ_t controls stochasticity: setting $\sigma_t = 0$ yields a fully deterministic process, enabling a one-to-one mapping between x_T (pure noise) and x_0. The DDPM reverse process can be recovered as a particular stochastic choice (if $\sigma_t = \sqrt{\beta_t}$) of the DDIM update, showing that DDIM generalizes DDPM sampling. DDIM deterministic trajectories (where $\sigma_t = 0$) form bijective maps between an image x_0 and a corresponding noise vector x_T. This enables inversion, where given a real image x_0, we can compute the unique x_T (the noise that would generate x_0 under DDIM) by running the reverse process forward (i.e., from step 0 to T). DDIM inversion has several applications such as precise reconstruction, image editing, and latent interpolation.

However, these pixel-space diffusion models are computationally expensive due to the high dimensionality of images. Latent diffusion models (LDMs) [5] address this by performing the diffusion process in a compressed latent space learned by a variational autoencoder (VAE), making both training and inference significantly more efficient. In practice, the underlying diffusion equations for LDMs remain the same as above, except that the image variable x is replaced with its latent equivalent z, obtained from the VAE encoder.

1.2. Attention Mechanisms in Diffusion Models

Diffusion models can be conditioned with external information such as class labels or text, so that the denoising U-Net not only removes noise but also steers the generation process [4]. The denoising network at each stage comprises of residual blocks that include self- and cross-attention mechanisms. At each denoising step, the noisy image x_t is passed through each stage of the U-Net's residual blocks, where it is transformed into feature maps. These feature maps are then sent into self- and cross-attention blocks. Inside self-attention, the feature maps are reshaped into a feature matrix $X \in \mathbb{R}^{N \times d}$ (with N spatial tokens and d feature dimensions). From this, Query (Q), Key (K), and Value (V) are computed as: $Q_{\text{self}} = XW_Q, K_{\text{self}} = XW_K, V_{\text{self}} = XW_V$, where W_Q, W_K, and W_V are learnable weight matrices. The self-attention is then defined as,

$$\text{SelfAttn}(X) = \text{softmax}\left(\frac{Q_{\text{self}} K_{\text{self}}^T}{\sqrt{d_{\text{self}}}} \right) V_{\text{self}}$$

This allows each location in x_t to attend to every other location, capturing long-range dependencies and global structure. Self-attention improves the spatial coherence of denoising and helps propagate information across the image grid. Similarly, inside the cross-attention block, the feature maps are reshaped into a feature matrix $X \in \mathbb{R}^{N\times d}$. Along with this, we have a conditioning matrix $H \in \mathbb{R}^{M\times d}$ (with M conditional tokens and d embedding dimensions) derived from the condition c. From these, Query (Q), Key (K), and Value (V) are computed as: $Q_{\text{cross}} = XW_Q, K_{\text{cross}} = HW_K, V_{\text{cross}} = HW_V$. The cross-attention is then defined as,

$$\text{CrossAttn}(X) = \text{softmax}\left(\frac{Q_{\text{cross}}K_{\text{cross}}^T}{\sqrt{d_{\text{cross}}}}\right)V_{\text{cross}}$$

In contrast to self-attention, where all Q, K, V come from the noisy image features, here, Queries (Q) come from noisy image features, while Keys (K) and Values (V) come from the conditioning signal. Through this, cross-attention integrates conditioning information into the network. Each query vector in Q corresponds to a spatial patch in the image, and cross-attention lets these spatial queries decide how strongly they align with the conditioning signal. This results in different spatial regions of the image responding to different parts of the condition, which acts as implicit localization. This cross-attention learns localization from the data, and this is where our disentangled latent conditions are particularly useful, as they reduce confusion between attributes and enhance the sharpness of localization.

1.3. Classifier Free Guidance

Classifier free guidance (CFG) [2] is a technique to guide the denoising U-Net and steer the generation towards the desired condition. During training, the condition c is randomly dropped with a certain probability, which enables the model to learn to predict noise both with conditioning $\epsilon_\theta(x_t, t, c)$ and without conditioning $\epsilon_\theta(x_t, t, \varnothing)$. At inference, both conditional and unconditional predictions are computed, and the final noise estimate is obtained by combining them as,

$$\epsilon_{\text{CFG}}(x_t, t, c) = \epsilon_\theta(x_t, t, \varnothing) + \gamma(\epsilon_\theta(x_t, t, c) - \epsilon_\theta(x_t, t, \varnothing))$$

where, γ is the guidance scale, which is a hyperparameter that amplifies the influence of the condition. A higher γ enforces stronger alignment with the condition, while a lower γ allows greater diversity. The combined noise ϵ_{CFG} is then used in the reverse process to generate the final image. Although the above formulation for Attention and CFG is expressed in terms of x_t, the same holds in latent space when using z_t.

1.4. Gradient Reversal Layer

The Gradient Reversal Layer (GRL) was introduced to adversarially remove unwanted information from learned features while still training the network for a primary task. Let $\text{GRL}_\lambda(\cdot)$ denote the layer with reversal weight $\lambda \geq 0$. Then, for a given input feature f, during the forward pass $\text{GRL}_\lambda(f)$ is simply f, while during the backward pass, $\frac{\partial \mathcal{L}}{\partial f}$ becomes $-\lambda \frac{\partial \mathcal{L}}{\partial \text{GRL}_\lambda(f)}$. This means, in the forward pass, the GRL acts as the identity function, and during the backward pass, it multiplies the incoming gradient by $-\lambda$, effectively turning a minimization objective into a maximization signal for upstream layers (*i.e.,* the layers preceding the GRL in the forward pass). This enables adversarial training via standard back propagation without the need for a separate optimizer [1].

In the context of our work, for each attribute-specific latent projection $\{z_i \mid i \in \text{attribute set, category}\}$, we have a direct classifier to which z_i is passed normally, and a set of reverse classifiers to which z_i is passed through the GRL. A compact formulation of the total loss for these latent classifiers is given as: $\mathcal{L}_{LC} = \sum_i \mathcal{L}_{\text{dir}} + \sum_i \sum_{j\neq i} \mathcal{L}_{\text{rev}}$. With the GRL in place, the gradient update for the parameters of the projection layer that produces z_i becomes:

$$\frac{\partial \mathcal{L}_{LC}}{\partial \theta_{z_i}} = \frac{\partial \mathcal{L}_{\text{dir}}}{\partial \theta_{z_i}} - \lambda \sum_{j\neq i} \frac{\partial L_{\text{rev}}}{\partial \theta_{z_i}}.$$

Here, $\frac{\partial \mathcal{L}_{\text{dir}}}{\partial \theta_{z_i}}$ encourages z_i to contain information relevant only to the i^{th} attribute, while $-\sum_{j\neq i} \frac{\partial \mathcal{L}_{\text{rev}}}{\partial \theta_{z_i}}$ discourages z_i from containing information about other attributes. This provides an efficient way to remove correlated attribute information from attribute-specific projections without requiring a separate adversarial training loop. The choice of λ controls the strength of disentanglement, in our experiments, we set $\lambda = 1$.

2. Working of Decision Module in FAE-Net

Algorithm 1 Attribute Editing using Preferred Attribute Class Label

```
 1: Input: $target_{index}, target_{label}$
 2: $z_A = [z_{\text{colorA}}, z_{\text{sleeveA}}, z_{\text{patternA}}, z_{\text{necklineA}}, z_{\text{categoryA}}]$
 3: $w_A = [w_{\text{colorA}}, w_{\text{sleeveA}}, w_{\text{patternA}}, w_{\text{necklineA}}]$
 4: if $w_A$[target_index] $\neq 0$ then
 5:     print "valid attribute manipulation"
 6:     $z_A[target_{index}]$ = random $z$ (where $z \in$ pool of $target_{label}$ latent representation)
 7: else
 8:     print "invalid attribute manipulation"
 9: end if
10: Output: $z_A$
```

The following algorithms illustrate the internal working of the decision module used during inference to prevent invalid attribute manipulations. The procedure differs slightly for the two cases considered. Algorithm 1 handles attribute

Algorithm 2 Attribute Editing using Source Image with Preferred Attribute

```
1:  Input: $target_{index}$
2:  $z_A = [z_{\text{colorA}}, z_{\text{sleeveA}}, z_{\text{patternA}}, z_{\text{necklineA}}, z_{\text{categoryA}}]$
3:  $w_A = [w_{\text{colorA}}, w_{\text{sleeveA}}, w_{\text{patternA}}, w_{\text{necklineA}}]$
4:  $z_S = [z_{\text{colorS}}, z_{\text{sleeveS}}, z_{\text{patternS}}, z_{\text{necklineS}}, z_{\text{categoryS}}]$
5:  $w_S = [w_{\text{colorS}}, w_{\text{sleeveS}}, w_{\text{patternS}}, w_{\text{necklineS}}]$
6:  if $w_A[target_{index}] \neq 0$ and $w_S[target_{index}] \neq 0$ then
7:      print “valid attribute manipulation”
8:      $z_A[target_{index}] = z_S[target_{index}]$
9:  else
10:     print “invalid attribute manipulation”
11: end if
12: Output: $z_A$
```

editing using a preferred attribute class label, while Algorithm 2 addresses attribute editing using a source image combined with the preferred attribute.

References

[1] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17 (59):1–35, 2016. 2

[2] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 2

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1

[4] Bingyan Liu, Chengyu Wang, Tingfeng Cao, Kui Jia, and Jun Huang. Towards understanding cross and self-attention in stable diffusion for text-guided image editing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7817–7826, 2024. 1

[5] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 1

[6] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 1