

FINAL REPORT MACHINE LEARNING OPERATIONS



JUDUL

Pembuatan dan Deployment Virtual - Try On Clothes yang Berbasiskan
CAT-VTON

Oleh:

Kelas: LA84

Angelino Delvin - 2802526271

Darren Makmur - 2802479741

Happy Victor J.K. - 2802540692

Machine Learning Operations (MLops)
Semester Ganjil 2025/2026

1. Introduction

1.1 Latar Belakang

Perkembangan teknologi digital telah mengubah perilaku konsumen secara fundamental, mendorong pergeseran masif dari belanja konvensional ke platform e-commerce . Meskipun menawarkan kemudahan akses dan variasi produk, belanja fashion secara daring memiliki kendala utama: ketiadaan pengalaman fisik untuk mencoba produk. Hal ini menyebabkan konsumen ragu akan kesesuaian ukuran dan gaya, yang berujung pada tingginya tingkat retur produk . Tingkat retur yang tinggi ini merugikan konsumen dari segi waktu dan menjadi beban operasional signifikan bagi pelaku usaha.

Salah satu solusi paling menjanjikan adalah teknologi Virtual Try-On (VTON) berbasis Artificial Intelligence (AI). Dengan memanfaatkan Computer Vision dan Deep Learning, teknologi ini memungkinkan visualisasi pakaian pada tubuh pengguna hanya melalui foto .

1.2 Rumusan Masalah

Tantangan teknis utama dalam pengembangan VTON adalah:

1. Menangani misalignment (ketidaksejajaran) antara foto pakaian katalog (datar) dengan postur tubuh pengguna yang kompleks.
2. Mempertahankan detail tekstur dan pola kain saat proses deformasi (warping) agar sesuai dengan lekuk tubuh.
3. Menyediakan antarmuka yang mudah digunakan bagi pengguna awam tanpa keahlian teknis.

1.3 Tujuan

Tujuan dari proyek ini adalah:

1. Merancang dan membangun purwarupa sistem VTON yang akurat dan realistik.
2. Mengimplementasikan model Deep Learning untuk segmentasi dan penyesuaian pakaian pada postur tubuh secara otomatis.
3. Mengembangkan antarmuka pengguna (UI) yang intuitif agar mudah dioperasikan.
4. Mengevaluasi akurasi visualisasi sistem sebagai tolok ukur keberhasilan.

1.4 Manfaat

- Bagi Konsumen: Meningkatkan kepercayaan diri saat belanja dan mengurangi risiko kesalahan pemilihan ukuran .
- Bagi Industri: Menekan angka retur produk dan meningkatkan konversi penjualan .

Bagi Lingkungan: Mengurangi jejak karbon akibat logistik pengembalian barang.

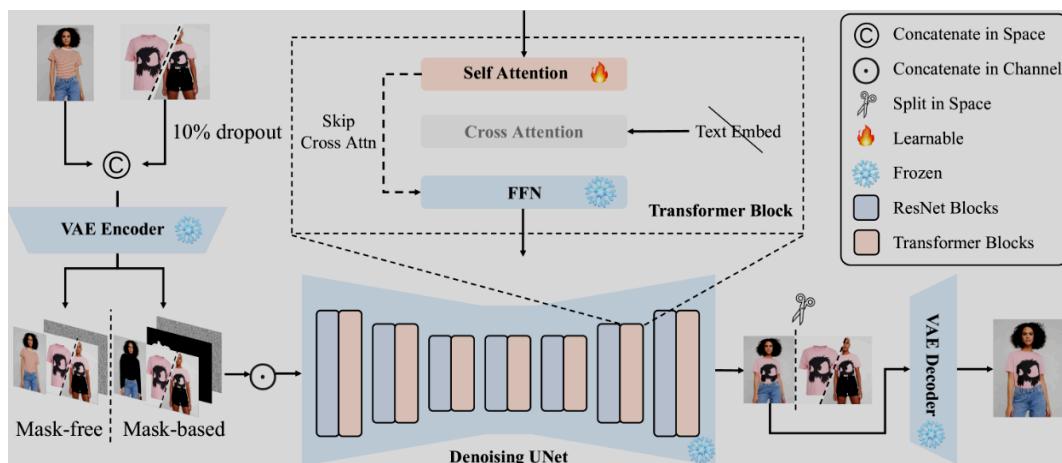
2. Dataset

Model CatVTON yang digunakan dalam sistem ini telah dilatih menggunakan gabungan dataset VITON-HD dan DressCode dengan total lebih dari 73.000 sampel citra berkualitas tinggi. Penggunaan dataset berskala besar ini memastikan model memiliki kemampuan generalisasi yang baik terhadap berbagai jenis pakaian (upper-body, lower-body, dan dresses).

3. Modelling

Sistem dibangun di atas kerangka kerja Latent Diffusion Model (LDM) dengan komponen utama sebagai berikut:

1. Autoencoder (AE): Mengompresi gambar input (orang dan pakaian) ke dalam latent space untuk menghemat memori GPU dan mempercepat inferensi .
2. Concatenation Input: Inovasi kunci di mana fitur laten dari gambar pakaian (garment), gambar target, dan mask digabungkan (concatenated) secara langsung pada dimensi channel.
3. Denoising UNet: Bertugas melakukan reverse diffusion, memprediksi dan menghilangkan noise pada area mask untuk mengisi area tersebut dengan fitur pakaian baru secara koheren .



Alur kerja sistem interface dirancang sebagai berikut:

1. Mulai: Pengguna membuka aplikasi web.
2. Input Pengguna: Pengguna mengunggah (1) foto diri dan (2) foto pakaian.
3. Preprocessing (Backend):
 - Sistem secara otomatis menghasilkan mask pada gambar pengguna (menghapus area pakaian lama).
 - Gambar diubah ukurannya (resize) sesuai resolusi model.

4. Proses Inti (CatVTON Pipeline):
 - Latent Encoding: Gambar pengguna, mask, dan pakaian dikonversi menjadi fitur laten menggunakan Auto Encoder.
 - Concatenation: Fitur-fitur laten tersebut digabungkan menjadi satu tensor input.
 - Denoising Process: UNet menjalankan proses difusi untuk mengisi area mask dengan detail pakaian baru berdasarkan noise yang diprediksi.
 - Decoding: Hasil latent dikembalikan menjadi gambar visual menggunakan Auto Decoder.
5. Tampilan Hasil: Gambar hasil try-on ditampilkan di antarmuka Streamlit ataupun Gradio ataupun Gradio.

4. Evaluation

Kualitas hasil sintesis citra pada CatVTON diukur menggunakan metrik standar industri seperti Fréchet Inception Distance (FID) untuk realisme dan Structural Similarity Index (SSIM) untuk kemiripan struktur. Secara komparatif, CatVTON mencatat skor FID 5.425 pada dataset VITON-HD, yang mengindikasikan tingkat realisme yang sangat tinggi dibandingkan metode difusi sebelumnya.

Methods	VITONHD						DressCode					
	Paired				Unpaired		Paired				Unpaired	
	SSIM ↑	FID ↓	KID ↓	LPIPS ↓	FID ↓	KID ↓	SSIM ↑	FID ↓	KID ↓	LPIPS ↓	FID ↓	KID ↓
DCI-VTON [11]	0.8620	9.408	4.547	0.0606	12.531	5.251	-	-	-	-	-	-
StableVTON [18]	0.8543	6.439	0.942	0.0905	11.054	3.914	-	-	-	-	-	-
StableGarment [37]	0.8029	15.567	8.519	0.1042	17.115	8.851	-	-	-	-	-	-
MV-VTON [36]	0.8083	15.442	7.501	0.1171	17.900	8.861	-	-	-	-	-	-
GP-VTON [40]	<u>0.8701</u>	8.726	3.944	<u>0.0585</u>	11.844	4.310	0.7711	9.927	4.610	0.1801	12.791	6.627
LaDI-VTON [23]	0.8603	11.386	7.248	0.0733	14.648	8.754	0.7656	9.555	4.683	0.2366	10.676	5.787
IDM-VTON [7]	0.8499	<u>5.762</u>	<u>0.732</u>	0.0603	<u>9.842</u>	<u>1.123</u>	0.8797	6.821	2.924	0.0563	<u>9.546</u>	<u>4.320</u>
OOTDiffusion [44]	0.8187	9.305	4.086	0.0876	12.408	4.689	<u>0.8854</u>	<u>4.610</u>	<u>0.955</u>	<u>0.0533</u>	12.567	6.627
CatVTON(Ours)	0.8704	5.425	0.411	0.0565	9.015	1.091	0.8922	3.992	0.818	0.0455	6.137	1.403

FID (Fréchet Inception Distance):

- Mengukur seberapa realistik gambar yang dihasilkan dengan membandingkan distribusi fitur gambar hasil generasi dengan gambar asli.
- Semakin rendah nilai FID, semakin baik (artinya gambar semakin realistik).
- CatVTON mencapai skor FID 5.425 pada dataset VITON-HD (paired setting), yang diklaim sebagai hasil terbaik dibanding metode lain saat itu.

SSIM (Structural Similarity Index):

- Mengukur kemiripan struktural antara gambar hasil generasi dan gambar asli (Ground Truth).
- Semakin tinggi nilai SSIM (mendekati 1), semakin baik (artinya struktur gambar sangat mirip dengan aslinya).
- Skor CatVTON pada VITON-HD: 0.8704.

KID (Kernel Inception Distance):

- Mirip dengan FID tetapi tidak bias (unbiased) dan bekerja lebih baik pada jumlah sampel yang sedikit.
- Semakin rendah nilai KID, semakin baik.
- Skor CatVTON pada VITON-HD: 0.411.

LPIPS (Learned Perceptual Image Patch Similarity):

- Mengukur perbedaan persepsi visual antara dua gambar (seberapa mirip gambar tersebut di mata manusia).
- Semakin rendah nilai LPIPS, semakin baik.
- Skor CatVTON pada VITON-HD: 0.0565.

5. Deployment

Tahap deployment merupakan fase krusial dalam implementasi sistem VTON, di mana model yang telah dikembangkan ditransformasikan menjadi layanan yang dapat diakses oleh pengguna akhir.

5.1 Platform Deployment

Sistem VTON dideploy menggunakan Hugging Face PRO sebagai platform utama. Platform ini dipilih karena menyediakan infrastruktur robust untuk aplikasi AI dengan kemampuan penskalaan otomatis, dukungan GPU optimal, integrasi API terstandarisasi, serta monitoring dan logging terintegrasi untuk memantau performa sistem secara real-time.

5.2 Permasalahan Deployment

- Optimasi Model Sebelum Deployment
Untuk mengatasi ketergantungan pada resource GPU yang mahal, dilakukan optimasi melalui pruning (menghilangkan neuron minimal kontribusi), knowledge distillation (transfer ke model compact), dan layer fusion (menggabungkan operasi berurutan).
- Deployment yang Memerlukan Pengeluaran
Model VTON memerlukan komputasi intensif untuk segmentasi, warping, dan rendering. Penggunaan GPU menjadi keharusan untuk mencapai waktu inferensi yang cukup, tanpa akses untuk GPU dalam Hugging Face PRO/Platform Deployment lainnya, kita tidak dapat mendeploy model kami dengan lancar.
- Fine Tuning dan Pencegahan Penyalahgunaan Setelah Deployment
Sistem dilengkapi safeguard berupa content moderation untuk filter input tidak pantas, watermarking digital untuk traceability, rate limiting per pengguna, dan user authentication untuk accountability.

5.3 Deployment Solution

- Arsitektur Modular
Struktur deployment dirancang dengan pemisahan komponen: file konfigurasi environment (.env), dependency management (requirements.txt), containerization (Docker) untuk portabilitas, dan model artifacts dengan versioning jelas untuk memudahkan rollback.
- Multi-Mode Model System

Sistem menyediakan tiga opsi generation dengan tujuan tersendirinya, berupa::

Upper: Generate bagian torso/atas tubuh.

Lower: Generate bagian kaki/bawah tubuh.

Overall: Generate upper dan lower bagian dari tubuh.

- Advanced Options

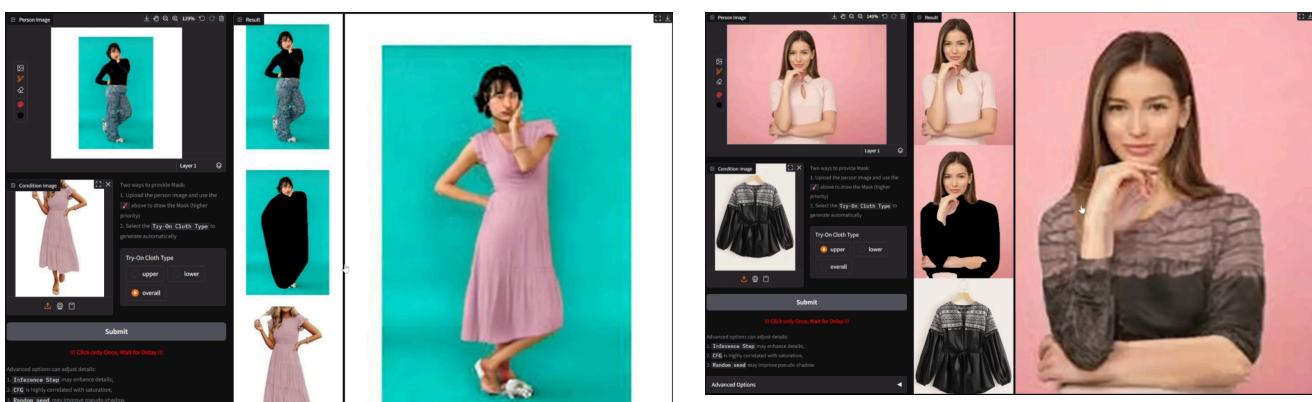
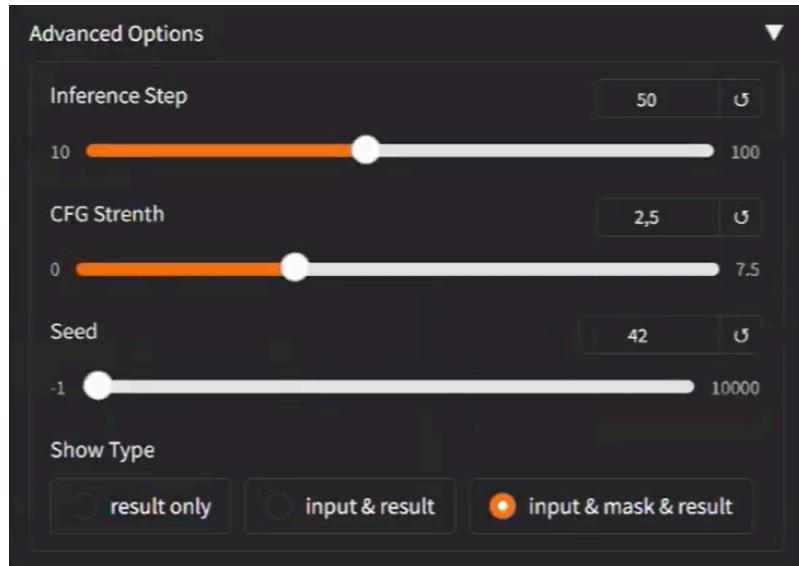
Model dilengkapi dengan fitur yang lebih advanced:

- Inference Steps (30-50 default): Mengontrol iterasi proses untuk balance kualitas dan kecepatan
- CFG Strength (2.5 default): Mengatur adherence model terhadap detail pakaian katalog
- Seed: Parameter reproducibility untuk debugging dan quality assurance
- Show Type: Opsi visualisasi (result only, input & result, atau input & mask & result).

5.4 Hasil Implementasi

Setelah mengetahui permasalahan yang ada pada model dan juga tahap deployment, kami mengimplementasikan solusi tersebut kepada model.

Antarmuka pengguna telah berhasil dibangun. Pengguna dapat memilih jenis pakaian (*upper/lower/overall*) dan melihat hasil di panel samping. Pada antarmuka pengguna, disediakan opsi 'Advanced Options' di mana pengguna dapat mengatur 'Inference Steps' untuk meningkatkan detail tekstur dan 'CFG' untuk menyesuaikan saturasi warna serta 'Seed' untuk mengatur tingkat ketetapan gambar (acak atau tidaknya hasil gambar)

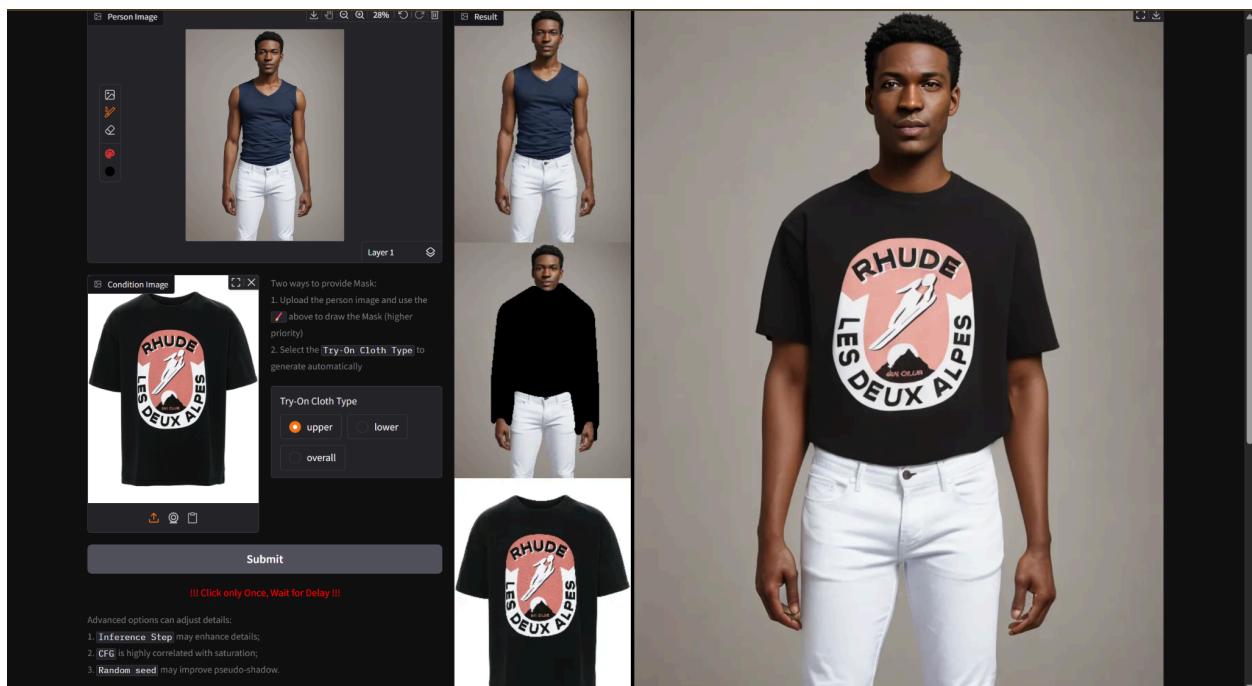


Secara visual, hasil pada menunjukkan bahwa tekstur logo pada pakaian berhasil dipertahankan tanpa distorsi berlebih, dan perbatasan antara kulit dan kain terlihat natural.

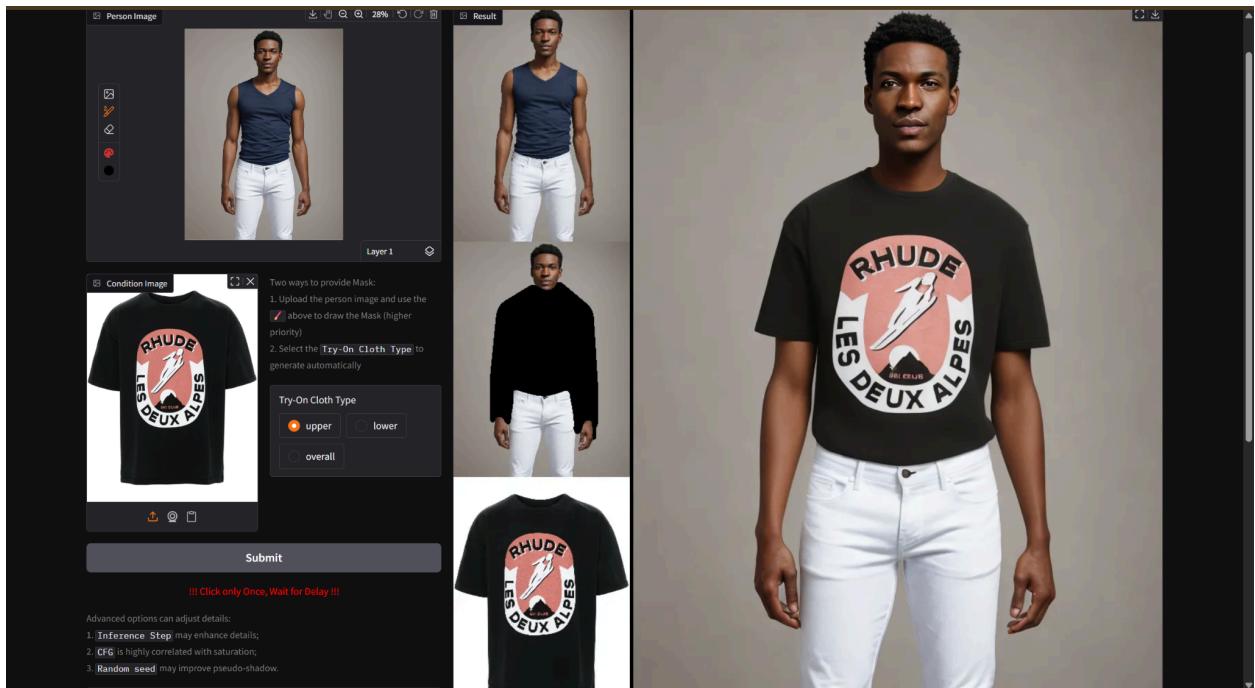
- Realisme: Detail tekstur, lipatan, dan bayangan pakaian dapat dibuat dengan sesuai.
- Koherensi: Kemulusan perpaduan antara pakaian virtual dan tubuh asli juga dapat dideteksi dari sedikitnya artefak (noise) yang terbentuk. Selain itu model berhasil menciptakan ulang bagian tubuh yang tertutup oleh pakaian (foto asli (lengan panjang) menjadi gambar yang di buat AI (lengan pendek))
- Validasi: inspeksi visual akan dilakukan untuk memastikan detail tekstur dan *warping* pakaian.

Berikut contoh dari penerapan *advanced options* :

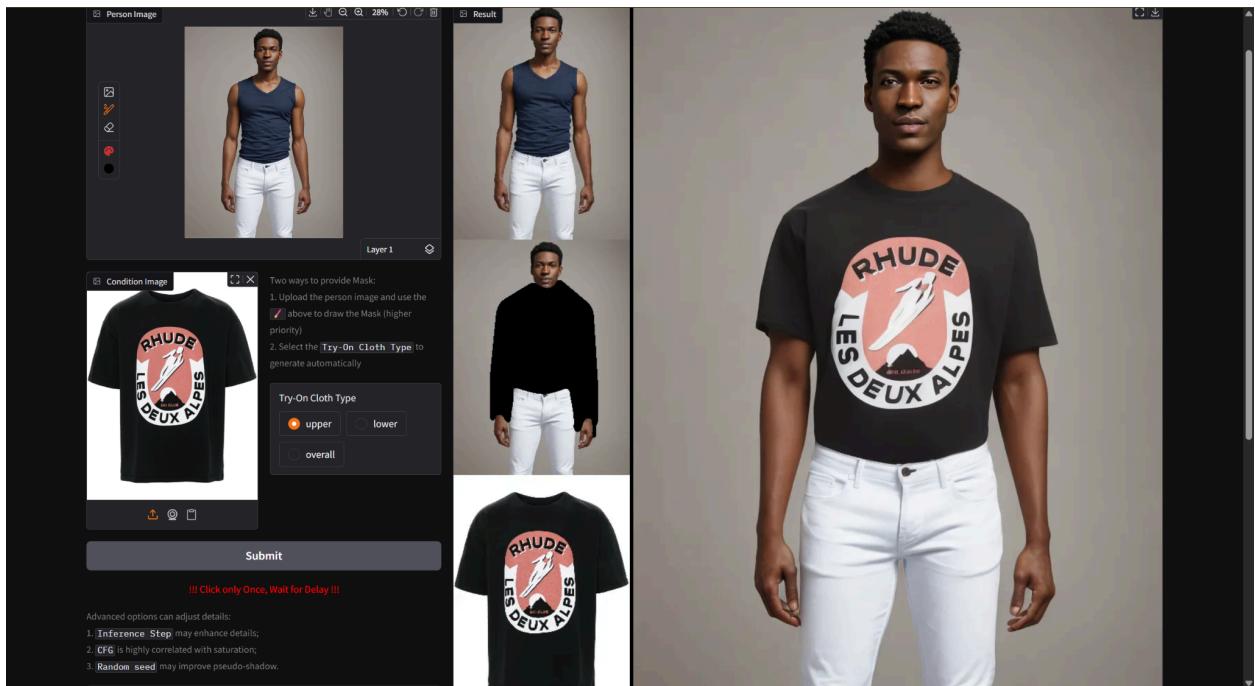
Default (inference 50, cfg 2,5 , seed 42)



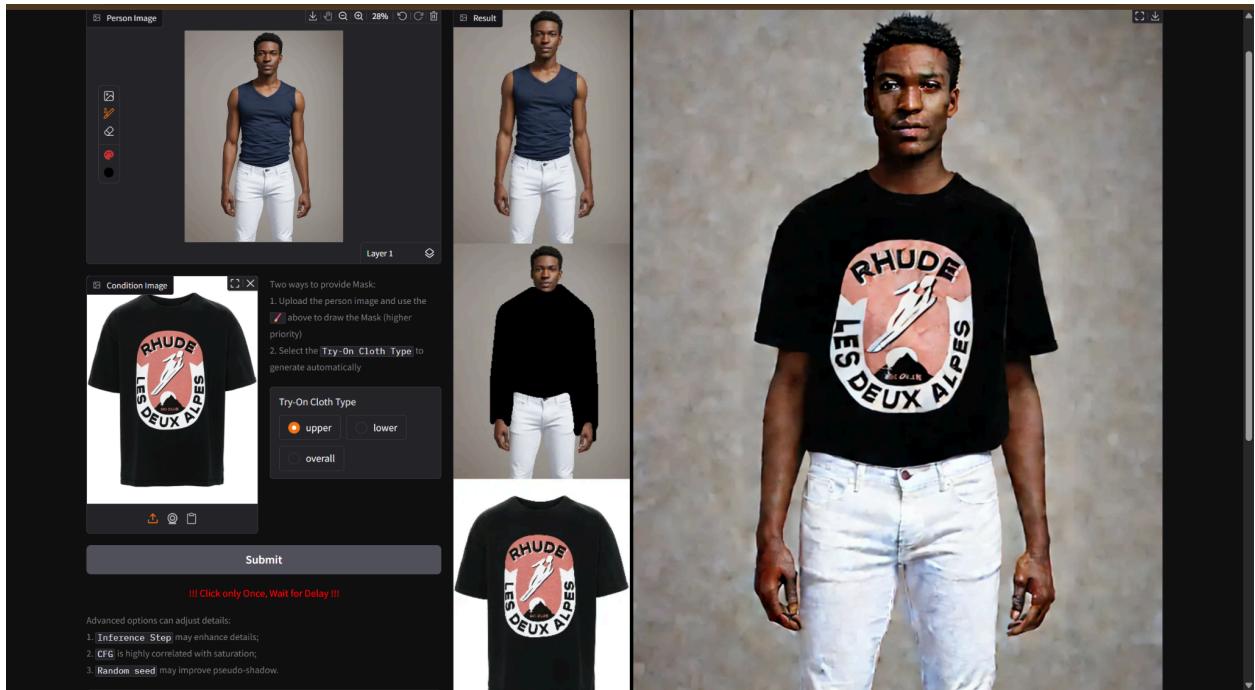
Inference = 10, cfg 2,5 , seed 42



Inference = 50, cfg 0,1 , seed 42

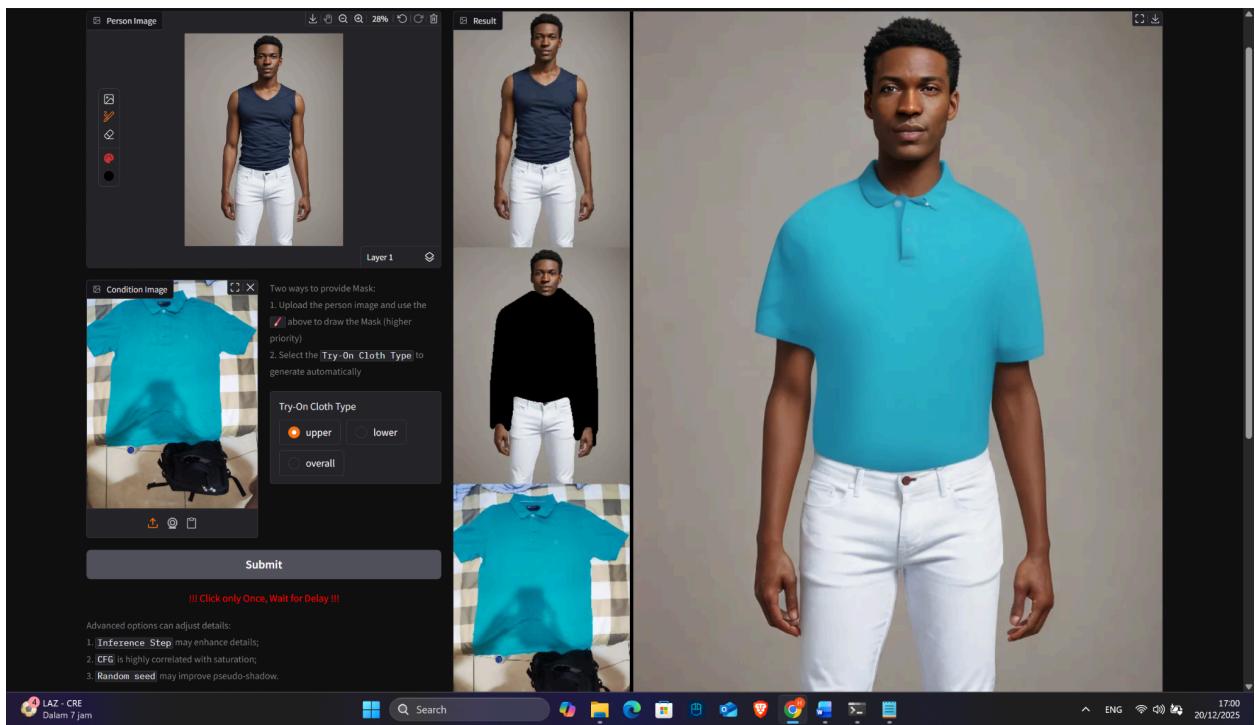


Inference = 50, cfg 7,5 , seed 42

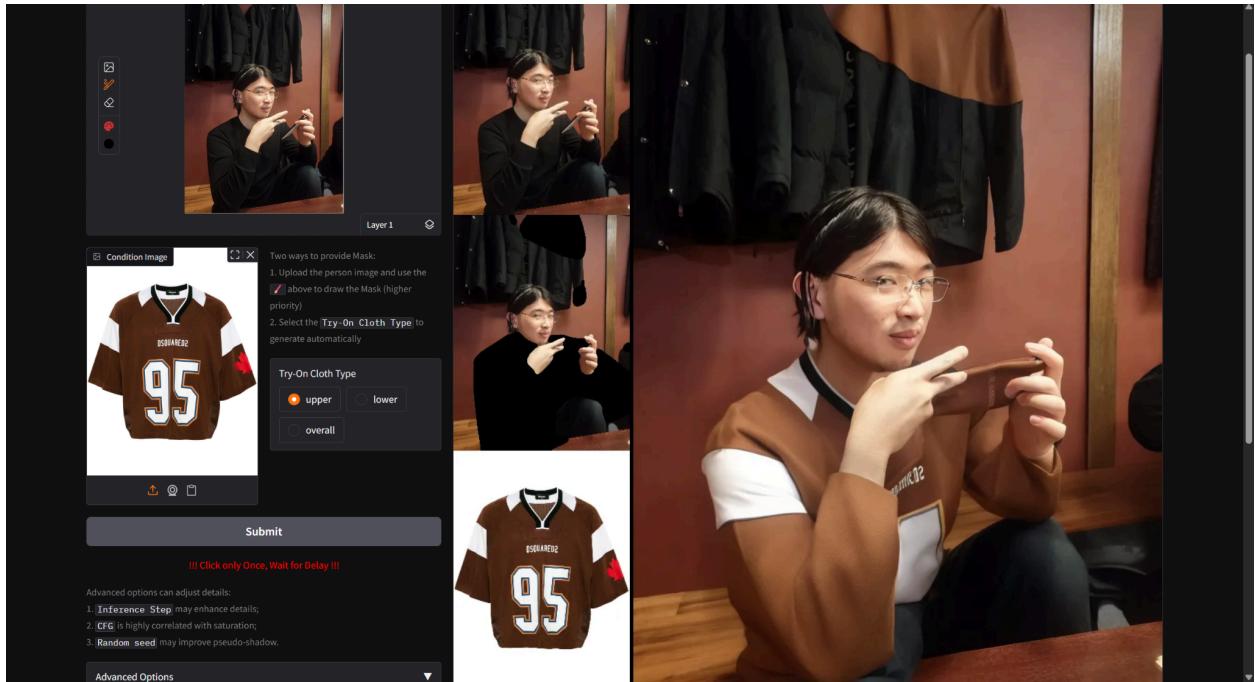


5.5 Penggunaan nyata (*real world usecase*) - default advanced options

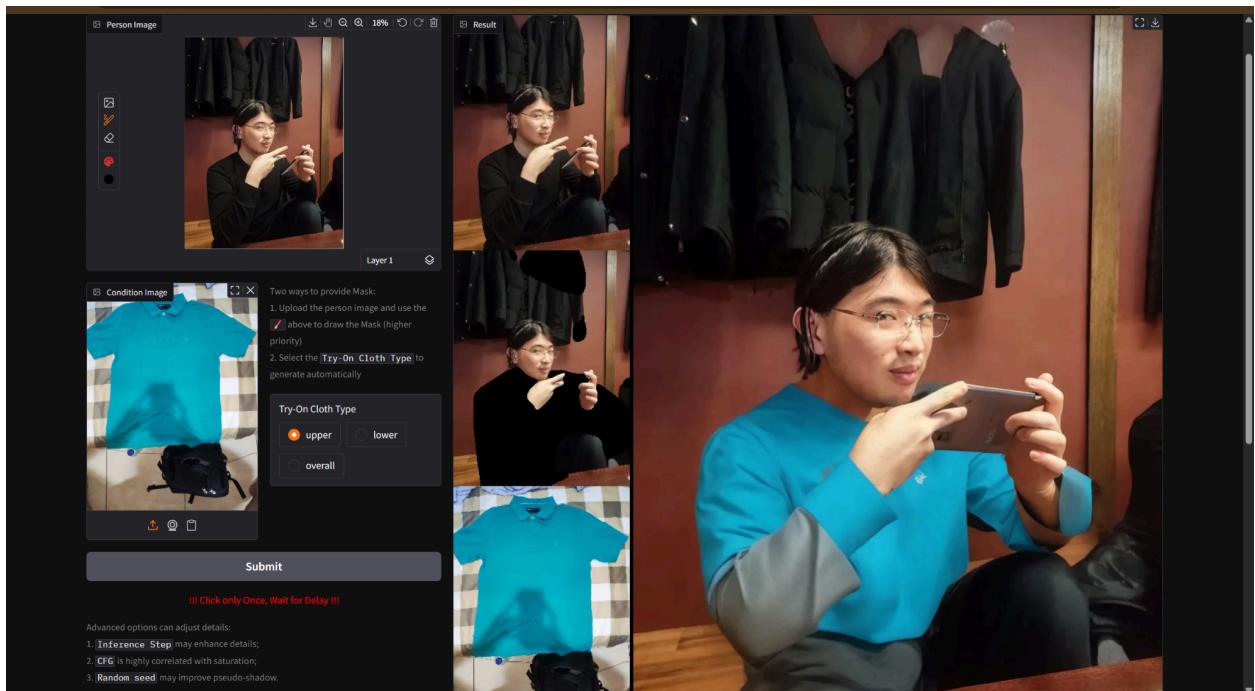
Baju dari foto hp



Model manusia dari foto hp



Model manusia dan baju dari foto hp



5.5 Fine tuning & hyper parameter tuning

Implementasi penanganan penyalahgunaan (NSFW) - stable diffusion safety checker

```

class StableDiffusionSafetyChecker(PreTrainedModel):
    def forward(self, clip_input, images):
        result = []
        batch_size = image_embs.shape[0]
        for i in range(batch_size):
            result_img = {"special_scores": {}, "special_care": [], "concept_scores": {}, "bad_concepts": []}

            # increase this value to create a stronger 'nsfw' filter
            # at the cost of increasing the possibility of filtering benign images
            adjustment = 0.0

            for concept_idx in range(len(special_cos_dist[0])):
                concept_cos = special_cos_dist[0][concept_idx]
                concept_threshold = self.special_care_embs_weights[concept_idx].item()
                result_img["special_scores"][concept_idx] = round(concept_cos - concept_threshold + adjustment, 3)
                if result_img["special_scores"][concept_idx] > 0:
                    result_img["special_care"].append((concept_idx, result_img["special_scores"][concept_idx]))
                adjustment = 0.01

            for concept_idx in range(len(cos_dist[0])):
                concept_cos = cos_dist[0][concept_idx]
                concept_threshold = self.concept_embs_weights[concept_idx].item()
                result_img["concept_scores"][concept_idx] = round(concept_cos - concept_threshold + adjustment, 3)
                if result_img["concept_scores"][concept_idx] > 0:
                    result_img["bad_concepts"].append(concept_idx)

            result.append(result_img)

        has_nsfw_concepts = [len(res["bad_concepts"]) > 0 for res in result]

        for idx, has_nsfw_concept in enumerate(has_nsfw_concepts):
            if has_nsfw_concept:
                if torch.is_tensor(images) or torch.is_tensor(images[0]):
                    images[idx] = torch.zeros_like(images[idx]) # black image
                else:
                    images[idx] = np.zeros(images[idx].shape) # black image

        if any(has_nsfw_concepts):
            logger.warning(
                "Potential NSFW content was detected in one or more images. A black image will be returned instead."
                "\nTry again with a different prompt and/or seed."
            )

        return images, has_nsfw_concepts

@torch.no_grad()
def forward(self, clip_input, images):
    return self.forward(clip_input, images)

```

Penyesuaian parameter terhadap pose tubuh manusia - optimalisasi pada pakaian bagian *upper* dan *overall*

```

def add_densepose_head_config(cfg: CN) -> None:
    add_densepose_head_cse_config(cfg)

def add_hrnet_config(cfg: CN) -> None:
    """
    Add config for HRNet backbone.
    """
    _C = cfg

    # For HigherHRNet w32
    _C.MODEL.HRNET = CN()
    _C.MODEL.HRNET.STEM_INPLANES = 64
    _C.MODEL.HRNET.STAGE2 = CN()
    _C.MODEL.HRNET.STAGE2.NUM_MODULES = 1
    _C.MODEL.HRNET.STAGE2.NUM_BRANCHES = 2
    _C.MODEL.HRNET.STAGE2.BLOCK = "BASIC"
    _C.MODEL.HRNET.STAGE2.NUM_BLOCKS = [4, 4]
    _C.MODEL.HRNET.STAGE2.NUM_CHANNELS = [32, 64]
    _C.MODEL.HRNET.STAGE2.FUSE_METHOD = "SUM"
    _C.MODEL.HRNET.STAGES = CN()
    _C.MODEL.HRNET.STAGES.NUM_MODULES = 4
    _C.MODEL.HRNET.STAGES.NUM_BRANCHES = 3
    _C.MODEL.HRNET.STAGES.BLOCK = "BASIC"
    _C.MODEL.HRNET.STAGES.NUM_BLOCKS = [4, 4, 4]
    _C.MODEL.HRNET.STAGES.NUM_CHANNELS = [32, 64, 128]
    _C.MODEL.HRNET.STAGES.FUSE_METHOD = "SUM"
    _C.MODEL.HRNET.STAGE4 = CN()
    _C.MODEL.HRNET.STAGE4.NUM_MODULES = 3
    _C.MODEL.HRNET.STAGE4.NUM_BRANCHES = 4
    _C.MODEL.HRNET.STAGE4.BLOCK = "BASIC"
    _C.MODEL.HRNET.STAGE4.NUM_BLOCKS = [4, 4, 4]
    _C.MODEL.HRNET.STAGE4.NUM_CHANNELS = [32, 64, 128, 256]
    _C.MODEL.HRNET.STAGE4.FUSE_METHOD = "SUM"

    _C.MODEL.HRNET.HRFPN = CN()
    _C.MODEL.HRNET.HRFPN.OUT_CHANNELS = 256

def add_densepose_config(cfg: CN) -> None:
    add_densepose_head_config(cfg)
    add_hrnet_config(cfg)
    add_bootstrap_config(cfg)
    add_dataset_category_config(cfg)
    add_evaluation_config(cfg)

def add_densepose_head_cse_config(cfg: CN) -> None:
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE = CN()
    # Dimensionality D of the embedding space
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.EMBED_SIZE = 16
    # Embedder specifications for various mesh IDs
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.EMBEDDERS = CN(new_allowed=True)
    # normalization coefficient for embedding distances
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.EMBEDDING_DIST_GAUSS_SIGMA = 0.01
    # normalization coefficient for geodesic distances
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.GEODESIC_DIST_GAUSS_SIGMA = 0.01
    # embedding loss weight
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.EMBEDDED_LOSS_WEIGHT = 0.6
    # embedding loss name, currently the following options are supported:
    # - EmbeddingLoss: cross-entropy on vertex labels
    # - SoftEmbeddingLoss: cross-entropy on vertex label combined with
    #   Gaussian penalty on distance between vertices
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.EMBEDDED_LOSS_NAME = "EmbeddingLoss"
    # optimizer hyperparameters
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.FEATURES_LR_FACTOR = 1.0
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.EMBEDDING_LR_FACTOR = 1.0
    # Shape to shape cycle consistency loss parameters:
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.SHAPE_TO_SHAPE_CYCLE_LOSS = CN({"ENABLED": False})
    # shape to shape cycle consistency loss weight
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.SHAPE_TO_SHAPE_CYCLE_LOSS_WEIGHT = 0.025
    # norm type used for loss computation
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.SHAPE_TO_SHAPE_CYCLE_LOSS_NORM_P = 2
    # normalization term for embedding similarity matrices
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.SHAPE_TO_SHAPE_CYCLE_LOSS_TEMPERATURE = 0.05
    # maximum number of vertices to include into shape to shape cycle loss
    # if negative or zero, all vertices are considered
    # if positive, random subset of vertices of given size is considered
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.SHAPE_TO_SHAPE_CYCLE_LOSS_MAX_NUM_VERTICES = 4936
    # Pixel to shape cycle consistency loss parameters:
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.PIX_TO_SHAPE_CYCLE_LOSS = CN({"ENABLED": False})
    # pixel to shape cycle consistency loss weight
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.PIX_TO_SHAPE_CYCLE_LOSS_WEIGHT = 0.0001
    # norm type used for loss computation
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.PIX_TO_SHAPE_CYCLE_LOSS_NORM_P = 2
    # map images to all meshed and back (If false, use only gt meshes from the batch)
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.PIX_TO_SHAPE_CYCLE_LOSS_USE_ALL_MESHES_NOT_GT_ONLY = False
    # Randomly select at most this number of pixels from every instance
    # if negative or zero, all vertices are considered
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.PIX_TO_SHAPE_CYCLE_LOSS_NUM_PIXELS_TO_SAMPLE = 100
    # normalization factor for pixel to pixel distances (higher value = smoother distribution)
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.PIX_TO_SHAPE_CYCLE_LOSS_PIXEL_SIGMA = 5.0
    # model ROI select
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.PIX_TO_SHAPE_CYCLE_LOSS_TEMPERATURE_PIXEL_TO_VERTEX = 0.05
    _C_MODEL_ROI_DENSEPOSE_HEAD_CSE.PIX_TO_SHAPE_CYCLE_LOSS_TEMPERATURE_VERTEX_TO_PIXEL = 0.05

```

Percobaan penggunaan model gabungan (CATVTON dengan flux) - namun tidak dimplementasikan secara penuh pada project akhir karena penggunaan kapasitas komputasi (spesifikasi *device*) yang kurang efesien (kami menilai penggunaan model *diffusion* lebih efesien dalam segi penyelesaian masalah dan hasil)

```

class FluxTryOnPipeline(
    DiffusionPipeline,
    FluxLatentLoaderMixin,
    FromSingleFileMixin,
    TextualInversionLoaderMixin,
):
    model_cpu_offload_seq = "transformer->vae"
    _optional_components = []
    _callback_tensor_inputs = ["latents"]

    def __init__(self, vae: AutoencoderKL, scheduler: FlowMatchEulerDiscreteScheduler, transformer: FluxTransformer2DModel):
        super().__init__()
        self.register_modules(vae=vae, scheduler=scheduler, transformer=transformer)

    self.vae_scale_factor = (
        2 ** (len(self.vae.config.block_out_channels) - 1) if hasattr(self, "vae") and self.vae is not None else 8
    )

    # Flux latents are turned into 2x2 patches and packed. This means the latent width and height has to be divisible
    # by the patch size. So the vae scale factor is multiplied by the patch size to account for this
    self.image_processor = VaeImageProcessor(vae_scale_factor=self.vae_scale_factor * 2)
    self.mask_processor = VaeImageProcessor(
        vae_scale_factor=self.vae_scale_factor + 2,
        vae_latent_channels=self.vae.config.latent_channels,
        do_normalize=False,
        do_binarize=True,
        do_convert_grayscale=True,
    )
    self.default_sample_size = 128

    self.transformer.remove_text_layers() # TryOnEdit: remove text layers

    @classmethod
    def from_pretrained(cls, pretrained_model_name_or_path, subfolder=None, **kwargs):
        transformer = FluxTransformer2DModel.from_pretrained(pretrained_model_name_or_path, subfolder="transformer")
        transformer.remove_text_layers()
        vae = AutoencoderKL.from_pretrained(pretrained_model_name_or_path, subfolder="vae")
        scheduler = FlowMatchEulerDiscreteScheduler.from_pretrained(pretrained_model_name_or_path, subfolder="scheduler")

class FluxAttnProcessor2_0:
    """Attention processor used typically in processing the SD3-like self-attention projections."""

    def __init__(self):
        if not hasattr(F, "scaled_dot_product_attention"):
            raise ImportError("FluxAttnProcessor2_0 requires PyTorch 2.0, to use it, please upgrade PyTorch to 2.0.")

    def __call__(self, attn: Attention, hidden_states: torch.FloatTensor, encoder_hidden_states: torch.FloatTensor = None,
                attention_mask: Optional[torch.FloatTensor] = None, image rotary emb: Optional[torch.Tensor] = None,
                ) -> torch.FloatTensor:
        batch_size, _, _ = hidden_states.shape if encoder_hidden_states is None else encoder_hidden_states.shape

        # 'sample' projections.
        query = attn.to_q(hidden_states)
        key = attn.to_k(hidden_states)
        value = attn.to_v(hidden_states)

        inner_dim = key.shape[-1]
        head_dim = inner_dim // attn.heads

        query = query.view(batch_size, -1, attn.heads, head_dim).transpose(1, 2)
        key = key.view(batch_size, -1, attn.heads, head_dim).transpose(1, 2)
        value = value.view(batch_size, -1, attn.heads, head_dim).transpose(1, 2)

        if attn.norm_q is not None:
            query = attn.norm_q(query)
        if attn.norm_k is not None:
            key = attn.norm_k(key)

        # the attention in FluxSingleTransformerBlock does not use `encoder_hidden_states`
        if encoder_hidden_states is not None:
            # 'context' projections.
            encoder_hidden_states_query_proj = attn.add_q_proj(encoder_hidden_states)
            encoder_hidden_states_key_proj = attn.add_k_proj(encoder_hidden_states)
            encoder_hidden_states_value_proj = attn.add_v_proj(encoder_hidden_states)

            encoder_hidden_states_query_proj = encoder_hidden_states_query_proj.view(
                batch_size, -1, attn.heads, head_dim
            ).transpose(1, 2)
            encoder_hidden_states_key_proj = encoder_hidden_states_key_proj.view(
                batch_size, -1, attn.heads, head_dim
            ).transpose(1, 2)

```

6. Reflection

Secara objektif, kami berhasil merancang purwarupa sistem *Virtual Try-On* menggunakan arsitektur CatVTON. Sistem ini mampu mensintesis citra pakaian pada tubuh pengguna dengan kualitas visual yang baik, menjawab kebutuhan akan pengalaman belanja daring yang lebih interaktif. Antarmuka berbasis Streamlit ataupun Gradio berfungsi dengan baik dalam memfasilitasi interaksi pengguna.

Pengembangan selanjutnya dapat difokuskan pada:

1. Optimasi kecepatan inferensi agar mendekati *real-time*.
2. Pengujian lebih lanjut pada variasi pose tubuh yang ekstrem.
3. Pengembangan versi aplikasi *mobile* untuk jangkauan pengguna yang lebih luas.
4. Penanganan Penyalahgunaan

Secara pribadi, masing-masing dari kami memiliki refleksi terhadap hasil dan juga proses dari proses penyelesaian project ini.

Angelino Delvin,

Proyek ini menjadi wadah pembelajaran bagi saya dalam menerapkan prinsip operasional pada pengembangan model CatVTON. saya menyadari bahwa keberhasilan sistem tidak hanya bergantung pada algoritma, tetapi juga pada koordinasi tim yang solid, pembagian peran yang jelas, serta komunikasi yang transparan. Kendala teknis yang muncul selama proses integrasi mendorong tim untuk lebih adaptif dalam pengambilan keputusan dan disiplin dalam menggunakan repository bersama. Pengalaman ini meningkatkan profesionalitas saya dalam mengelola proyek teknologi secara end-to-end, memastikan setiap modul siap digunakan oleh pengguna akhir dengan risiko kegagalan yang minimal.

Darren Makmur,

Proyek Virtual Try-On berbasis CatVTON ini menjadi proses pembelajaran yang berharga dalam bekerja secara kolaboratif pada lingkup MLOps. Setiap tahap, mulai dari memahami arsitektur model, pengolahan dataset, hingga deployment menuntut koordinasi yang baik, pembagian peran yang jelas, serta komunikasi yang konsisten dalam tim. Tantangan teknis seperti optimasi model, keterbatasan resource, dan pengujian sistem membantu tim menyadari pentingnya repository, troubleshooting yang sistematis, dan pengambilan keputusan yang melibatkan seluruh anggota. Melalui proses ini, tim tidak hanya mengembangkan produk berbasis AI, tetapi juga meningkatkan profesionalitas dalam mengelola proyek teknologi secara end-to-end, dari tahap riset hingga siap digunakan oleh pengguna nyata.

Happy Victor J.K.,

1. Belajar Teknologi Baru yang Lebih Canggih

Jujur saja, membuat AI yang bisa memakaikan baju ke foto orang itu susah. Tapi lewat proyek ini, saya belajar pakai metode baru bernama CatVTON. Ternyata, metode ini lebih pintar karena bisa langsung menggabungkan gambar baju dan badan tanpa ribet, dan hasilnya bajunya terlihat pas dan teksturnya tetap terlihat.

2. Tantangan: Komputer "Ngebut" dan Error Program

Bagian paling menantang adalah saat coding. Model AI ini ternyata butuh memori komputer (VRAM) yang sangat besar. Selain itu, saya sering pusing karena programnya sering error gara-gara versi software yang tidak cocok satu sama lain (library). Di sini saya benar-benar belajar sabar dan teliti buat memperbaiki error satu per satu sampai programnya jalan lancar. Dan untuk mempermudha implementasi selanjutnya saya juga membuat virtual environment juga.

3. Membuat Aplikasi yang Gampang Dipakai

Saya tidak mau teknologi canggih ini hanya jalan di kode program saja. Jadi, saya buatkan tampilan web sederhana pakai Streamlit/Gradio. Tujuannya supaya orang awam tinggal upload foto badan dan foto baju, lalu klik tombol, dan hasilnya keluar. Tantangannya adalah mengatur supaya prosesnya tidak terlalu lama menunggu.

4. Kemampuan Menjelaskan

Selain coding, saya juga bertugas membuat laporan dan video presentasi. Ini melatih saya untuk tidak hanya teknis, tapi juga harus bisa menjelaskan hal rumit dengan bahasa yang dimengerti orang lain.

7. References

1. Chong, Z., Hu, X., Cao, H., Zhang, X., & Ren, P. (2024). *CatVTON: Concatenation is all you need for virtual try-on with diffusion models*. arXiv.
2. Chong, Z., et al. (2024). *CatVTON GitHub Repository*. GitHub.
3. happyvictor-Vesper. (2024). *DL---Try-On-CLothes Source Code*. GitHub.
4. Future Thinker @Benji. (2024). *CatVTON In ComfyUI*. YouTube.
5. happyvictor-Vesper. (2025). *MLops-repo*. GitHub.
<https://github.com/happyvictor-Vesper/MLops-repo>