**Documentation Report**

#### ***1. Model Overview***

For the prototype, I selected the following models:

1. **Stable Diffusion** (stabilityai/stable-diffusion-2) for **Text-to-Image Generation**.
2. **Text-to-Video Generation Model** (damo-vilab/text-to-video-ms-1.7b).
3. **Vision Encoder Decoder Model** (nlpconnect/vit-gpt2-image-captioning) for **Image Captioning (Image-to-Text)**.

Each model serves a distinct purpose for image and video generation or captioning based on input text or images.

#### ***2. Reason for Model Selection***

1. **Stable Diffusion for Text-to-Image Generation**:
   1. **Stable Diffusion** is a state-of-the-art diffusion model for generating high-quality images from text prompts.
   2. It is highly customizable, allowing control over the guidance scale, the number of inference steps, and seed-based randomization, making it an excellent choice for generating diverse images based on text inputs.
   3. **Chosen for:** Robust results and flexibility in generating detailed images across various contexts.
2. **Damo-vilab Text-to-Video Model**:
   1. **Damo-vilab's text-to-video model** is one of the top-performing models for generating short videos from a text prompt. It uses a transformer-based architecture and diffusion techniques to model the visual dynamics required to synthesize coherent video clips.
   2. **Chosen for:** Ability to create high-quality, dynamic video sequences from textual descriptions, supporting video-based content generation.
3. **Vision Encoder-Decoder Model for Image Captioning**:
   1. The **Vision Encoder-Decoder model** combines a vision transformer (ViT) with a GPT-2-based text generator, allowing it to generate descriptive captions for images.
   2. **Chosen for:** Its efficiency and accuracy in producing human-like captions for images, which is crucial for applications that involve multimodal content understanding.

#### ***3. Integration into the Prototype***

The models were integrated into a unified **Streamlit** interface where users can select the task they want to perform (text-to-image, text-to-video, or image-to-text).

Here’s how each model was integrated:

1. **Stable Diffusion (Text-to-Image Generation) Integration**:
   1. I used the diffusers library to load the Stable Diffusion model and provided the necessary configurations for inference, such as num\_inference\_steps and guidance\_scale.
   2. The generated images are displayed within the Streamlit interface using matplotlib, and users can download them as .png files.
   3. Translation is supported by integrating the **GoogleTrans** API, enabling image generation based on multilingual inputs.

**Code Example:**

*image\_gen\_model = StableDiffusionPipeline.from\_pretrained(*  
 *CFG.image\_gen\_model\_id,*  
 *torch\_dtype=torch.float16,*  
 *revision="fp16",*  
 *use\_auth\_token=os.environ.get('YOUR\_HUGGINGFACE\_TOKEN')*  
*).to(CFG.device)*

1. **Damo-vilab Text-to-Video Integration**:
   1. The diffusers library is used to load the video generation pipeline, and the export\_to\_video utility is employed to synthesize video frames from text prompts.
   2. These video frames are rendered into a video format (mp4) using imageio and displayed in the Streamlit app using HTML and base64 encoding.

**Code Example:**

*pipe = DiffusionPipeline.from\_pretrained("damo-vilab/text-to-video-ms-1.7b", torch\_dtype=torch.float16)*  
*video\_frames = pipe(prompt, num\_frames=num\_frames).frames*  
*display\_video(video\_frames, fps=10)*

1. **Vision Encoder Decoder (Image-to-Text) Integration**:
   1. The VisionEncoderDecoderModel was integrated via the Hugging Face transformers library.
   2. The model processes uploaded images, generating captions which are then optionally translated using the **GoogleTrans** API, allowing support for multilingual outputs.

**Code Example:**

*model = VisionEncoderDecoderModel.from\_pretrained("nlpconnect/vit-gpt2-image-captioning")*  
*pixel\_values = feature\_extractor(images=images, return\_tensors="pt").pixel\_values*  
*output\_ids = model.generate(pixel\_values, \*\*gen\_kwargs)*

#### ***4. Why These Models?***

These models were selected based on their performance and ease of integration:

* **Stable Diffusion** excels in text-to-image generation with flexibility in control.
* **Damo-vilab’s text-to-video model** offers one of the best solutions for text-driven video creation.
* The **Vision Encoder-Decoder** model ensures accurate caption generation for images, making it ideal for image-to-text translation.

By combining these models, the prototype can handle various multimodal tasks effectively, offering an intuitive and seamless user experience.

#### ***5. Prototype Features***

The interface offers the following features:

1. **Text-to-Image Generation**: Allows users to input text in any language, which is translated to English before generating an image.
2. **Text-to-Video Generation**: Generates video clips based on user input, providing high-quality, dynamic visuals.
3. **Image-to-Text Generation**: Users can upload images, and the system will generate captions in the selected language.

The use of **Streamlit** as a frontend makes the interface interactive and user-friendly, allowing easy switching between tasks and real-time content generation.

### **6. Deployment**

#### ***6.1 Deployment Attempts***

The deployment of the prototype was attempted on two popular platforms: **Hugging Face Spaces** and **AWS (Amazon Web Services)**. Both platforms are commonly used for deploying machine learning models and applications due to their robust infrastructure and ease of use. However, each of these platforms has its limitations that affected the deployment of this specific prototype.

1. **Hugging Face Spaces**:
   1. Hugging Face Spaces is a free hosting platform designed for showcasing machine learning models and applications.
   2. However, it has a strict limit of 1 GB for the total space allowed per space. Given the size of the models used in the prototype (especially the text-to-image and text-to-video generation models), this limitation posed a significant challenge.
   3. Due to the large sizes of the required dependencies and model weights, I was unable to deploy the prototype on Hugging Face Spaces.
2. **AWS (Amazon Web Services)**:
   1. AWS offers a more flexible and scalable environment for deployment. However, the cost of hosting can be prohibitive for projects that need to be free of charge.
   2. Similar to Hugging Face Spaces, AWS also encountered space constraints when deploying the application with the required dependencies and model weights.
   3. The combination of high storage requirements and the need for efficient resource management limited my ability to utilize AWS effectively for this project without incurring costs.

#### ***6.2 Workaround for Deployment***

Despite the challenges faced during the deployment process, I have documented the steps required to run the prototype locally in both the **notebook** and the **README** file. This allows users to replicate the environment and run the application on their local machines, ensuring they can experience its functionality in real-time.

* **Local Setup Instructions**:
  + The README file includes a comprehensive guide on how to set up the required environment, install dependencies, and run the Streamlit application.
  + Users are instructed on how to download the necessary models and place them in the appropriate directories, allowing the application to function without being constrained by cloud platform limitations.