FFMPEG

Open source command line process video,audio ,files etc…checks if ffmpeg is there or not by command line $ffmpeg –version.To create video from sequence of images .specify input images ,output file.Suppose there are same format image like happy.1 ,happy.2 jpeg then can use ffmpeg with –i happy%d format to load images.This will search for lowest digit in the formal search images and places it before.Incerement number by 1 after loading particular images. –start_number_n (start image number)

-framerate 1 this sets frames 1 per second

-c:v lib x264 –r 30 this gives video frame rate fixed to 30 frames per second.It chooses H.264 for encoding.H.264 compress video data,making it suitable for storage,transmission and straming over various networks.yuv420p is added when output is H.264 to increase the compatibility of video.*jpg format matching files will be sorted according to LC-COLLATE(global format).LC-COLLATE used when images are not in sequential order. can use “cat” to pipe to ffmpeg.”cat” concatenate multiple files to ffmpeg and by piping can process the concatenated data.shortest sets output length video size shortest than the input files.Images can be in different sizes, so we use scale with pad to fit image into a specific size.fade ,d sets duration of fade and st states when fades will start.

Shotstack API:

It is nothing but creating a slideshow video.Official docs ‘peg’ encode ,decode,transcode,mux, demux, stream and filters.Major drawback is steep learning curve.It don’t give access to take full charge of features.

FFMPEG:

CLI should use when take ffmpeg but most of users got accustomed to GUI’s.Understanding technical terms : codecs,containers,bitrates,filters.Limited documentation.

To overcome:

Use GUI frontends.

LORA (Low Rank Adaptation):

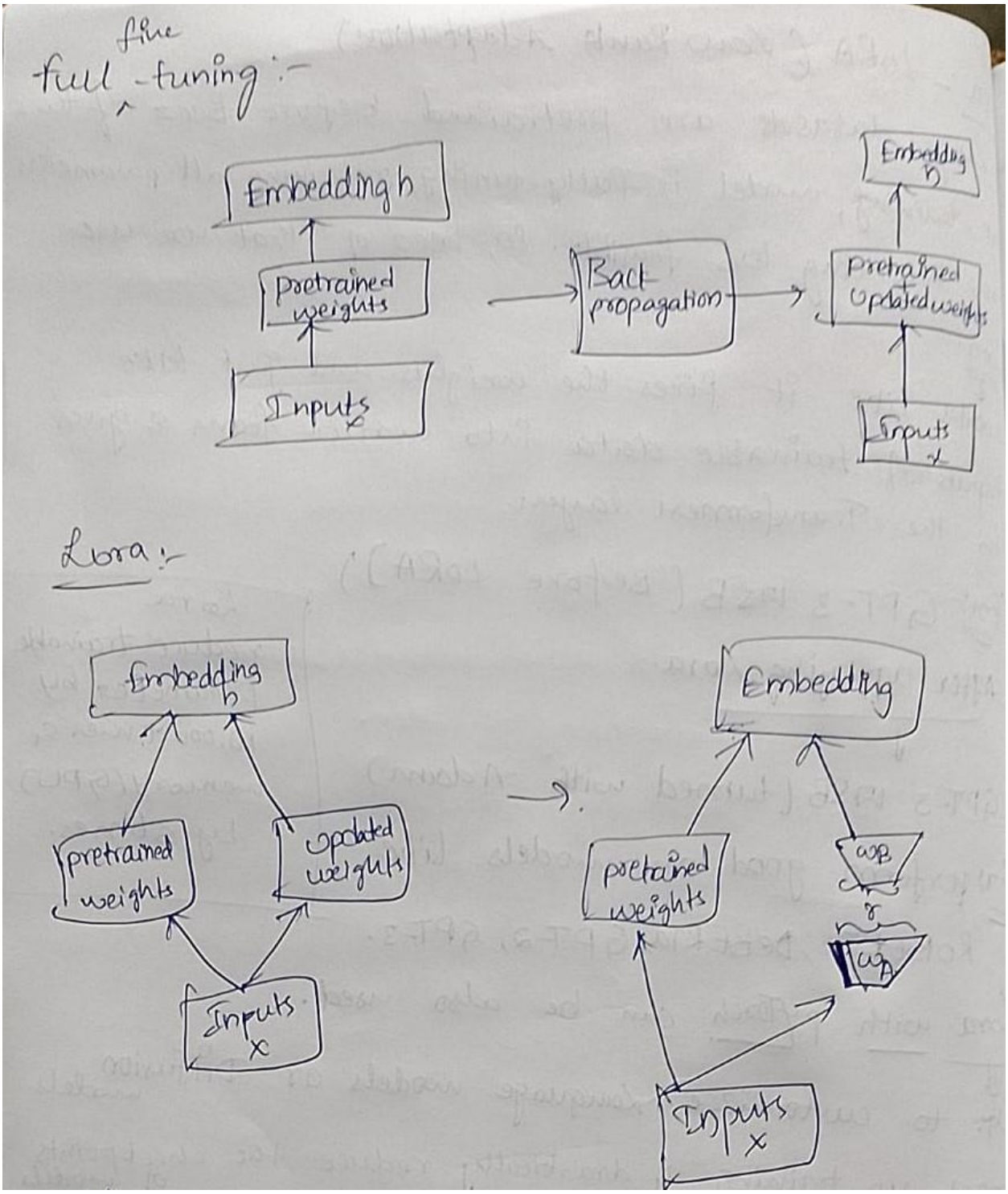
Large datasets are pretrained before. Because of this pre_training ,model is fully_tuning,retaining all parameters and becoming less feasible.So,because of that we use.Works like it fixes the weights and just take inputs of trained data into matrix form and gives to the transformers layers.

Ex: GPT-3 175B(Before Lora)

After applying lora GPT-3 175B(Tuned with Adam)

Example : Perform good on models like ROBERTa,DeBERTa,GPT-2,GPT-3.

Lora with pytorch can also be used.Lora helps to customize Language models or diffusion models.Speeds up training and drastically reduce the checkpoints of models by training few parameters compared to base model ,while processing full-tuning.



It mainly focused on “attention weights”. Attention weights mainly has query and values.Without using Actual weight matrices of models we can only use those tiny injections i.e;small matrices..No need to use big optimizers.

Main use : Reduces storage memory for example GPT-3.

Spatial Transformers:

These are used to rotate,change angle of the images and also to make images stand in proper way for making user observations good.Here we use Grid generator. Grid generator typically refers to a component or function used in spatial transformation operations, particularly in the context of image warping or geometric transformations. These transformations are often used in tasks like image registration, augmentation, or in the context of spatial transformer networks.

The grid generator is responsible for generating a sampling grid, which is a set of coordinates that define how the pixels from the input image should be sampled and transformed to produce the output image. The grid generator plays a crucial role in defining the spatial transformation applied to the input image.

The grid generator is responsible for creating such grids based on the desired spatial transformation. It allows for flexibility in specifying how the input pixels should be rearranged to form the output image, enabling a variety of geometric transformations. This is especially useful in scenarios like image registration or when we want to apply specific spatial transformations to images during training, such as in data augmentation for deep learning models.

We also have sampler it is an object responsible for specifying the strategy used to draw samples from a dataset during training. The sampler determines the order in which the data samples are accessed, and it can influence how batches are formed during each iteration through the dataset.

Some common types of samplers are:

K Sampler

Random Sampler

Sequential Sampler

Subset Sampler

Weighted Sampler

Batch Sampler

Temporal Transformers:

It is a powerful model for multihorizon and multivariate time series forecasting use cases.Sequence of data i.e; it predicts future data by taking outputs of prediction results of past data. Example is wheather forecasting.GRN(Gate Residual Network) ,they just predicts or they are responsible to carry the data or skip the data.

Static covariate encoders:

Learn context vectors from static metadata and inject them at different locations of TFT model. Models like ViT (Vision Transformer) and its extensions have been successful in handling visual data. We can use these models to process video frames.Combine a Vision Transformer with a language model or temporal transformer to handle both the visual and temporal aspects of the task. Architectures like GPT (Generative Pre-trained Transformer) can be adapted to generate text sequences that correspond to video frames.Might need to extend the model to handle temporal dependencies explicitly, considering the order of frames in a video.

Parameters of stable diffusion.

1. images (List[PIL.Image.Image] or np.ndarray) — List of denoised PIL images of length batch_size or numpy array of shape (batch_size, height, width, num_channels). PIL images or numpy array present the denoised images of the diffusion pipeline.
2. nsfw_content_detected (List[bool]) — List of flags denoting whether the corresponding generated image likely represents “not-safe-for-work” (nsfw) content, or None if safety checking could not be performed.
3. images (List[PIL.Image.Image] or np.ndarray) — List of denoised PIL images that were flagged by the safety checker any may contain “not-safe-for-work” (nsfw) content, or None if no safety check was performed or no images were flagged.
4. applied_safety_concept (str) — The safety concept that was applied for safety guidance, or None if safety guidance was disabled.
5. prompt (str or List[str]): The text prompt or prompts provided to guide the image generation. This is what influences the content or style of the generated image.
6. height, width (int, optional): The dimensions (height and width) in pixels of the generated image.
7. num_inference_steps (int, optional): The number of denoising steps or iterations during the image generation process. More steps generally lead to higher-quality images but may increase the computational time.
8. guidance_scale (float, optional): A parameter that influences the balance between the influence of the text prompt and the quality of the generated image. Higher values prioritize fidelity to the text prompt but may result in lower image quality.
9. negative_prompt (str or List[str], optional): A prompt or prompts that guide what should not be included in the image generation. This is ignored when not using guidance.
10. num_images_per_prompt (int, optional): The number of images to generate per prompt.

11. eta (float, optional): A parameter corresponding to the eta (η) from the DDIM paper. It applies to a specific scheduler and affects the inference process.
12. generator (torch.Generator or List[torch.Generator], optional): A torch.Generator to make the generation process deterministic.

13. latents (torch.FloatTensor, optional): Pre-generated noisy latents sampled from a Gaussian distribution. These latents are used as inputs for image generation and can be used to tweak the same generation with different prompts.
14. output_type (str, optional): The format of the generated image output, either PIL.Image or np.array.
15. return_dict (bool, optional): Whether to return a StableDiffusionPipelineOutput instead of a plain tuple.
16. Callback (Callable, optional): A function that is called at specified steps during inference. Useful for monitoring or modifying the generation process.
17. callback_steps (int, optional): The frequency at which the callback function is called.
18. sld_guidance_scale (float, optional): Safety guidance scale. If less than 1, safety guidance is disabled.
19. sld_warmup_steps (int, optional): Number of warmup steps for safety guidance. Applied only for diffusion steps greater than sld_warmup_steps.
20. sld_threshold (float, optional): Threshold that separates appropriate and inappropriate images based on safety guidance.
21. sld_momentum_scale (float, optional): Scale of the safety guidance momentum added at each diffusion step. If 0, momentum is disabled.
22. sld_mom_beta (float, optional): Defines how safety guidance momentum builds up. Indicates how much of the previous momentum is kept during warmup.