Longyi Zhang (u3641195)
COMP7606B - Deep Learning

Assignment 2 Report

Introduction

For this assignment we are completing a small UNet-based Denoising Diffusion Probabilistic Model (DDPM) and its upsampling block to generate images of handwritten digits based on the MNIST dataset. We are specifically tasked to:

- Implement UNet upsampling process and its architecture.
- Implement linear variance scheduling in addition to the default cosine schedule.
- Define the model forward diffusion pass.

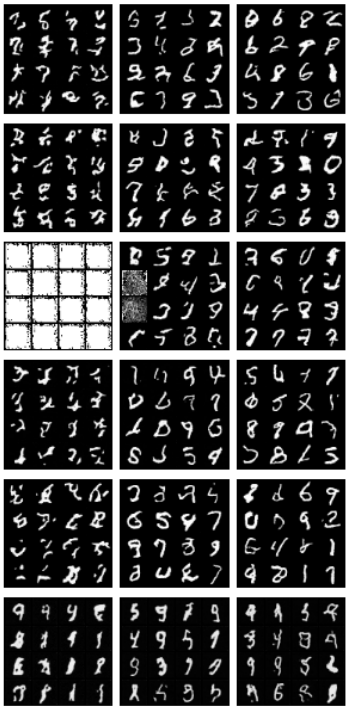
We then augment the unconditional model to enable conditional generation of given digits, as well as comparing its performance on unconditional generation in different configurations.

Method

We defined a baseline model in line with the parameters provided in the code base with cosine variance schedule, data normalisation with both mean and standard variation at 0.5, clipping to [-1,1] during reverse diffusion, 1000 samping steps, 64 base layers, and 256 timestep embedding layers. We planned to explore certain model parameters of interest and their impact on performance.

We first use a **linear variance schedule** in lieu of the cosine schedule. Our implementation is identical to that specified in the DDPM paper, with starting value 1e-4 and ending value 0.02. We then experiment with the omission of **clipping** and **normalisation** during preprocessing. Finally, we vary the number of **sampling steps** to examine its effect on training loss and the generated image quality. For this experiment set we selected 100, and 10, 1 and 2 magnitudes smaller than the provided rate.

To enable conditional generation, we proposed an implementation closely resembling the handling of timesteps. To leverage the existing UNet architecture, we combined the time and label embeddings instead of incorporating them separately. We additionally enabled MNIST dataset labels to be passed into the diffusion model and extend the arguments to allow conditional generation of target digits specified in the input.



Top to bottom: baseline, linear schedule, no clip, no normalisation, 100 steps, 10 steps. At epoch 10, 20, and 30

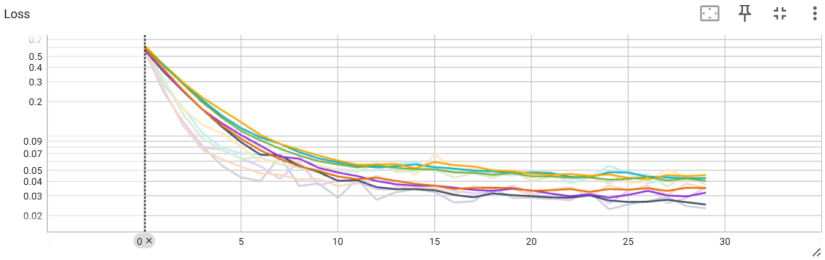
Experiments & Analysis

6 experiments were performed corresponding to the above variations using the unconditional generative model. Each model was trained for 30 epochs with evaluation images generated at every 10th epoch. The implementation and results of the conditional model is detailed afterwards, with reference to this demonstrative model.

Results

Compared to the base model, **linear variance schedule** yielded the most significant improvement in training loss throughout the run. Rather than the scheduler function itself, the result likely stems from the much lower variance range ([1e-4,0.02] against [0,1]). We see the consequence of this low ceiling at higher epochs, where products appear more homogeneous and less realistic compared to baseline.

Despite negligible impact on convergence, the **absence of clipping** essentially rendered the model unusable. This specific instance generated near-max values across all images, likely due to interference from out-of-range values, at epoch 10 and occasionally failed to denoise the image at epoch 20. By epoch 30, despite no longer producing artefacts, it generated subpar results compared to other conditions. In contrast, doing away with zero-centred **normalisation** during preprocessing seemingly had little impact on generative quality, at least on the simple, monochromatic MNIST dataset.



Run	Min	Max	Start Value	End Value	△Value	△%	Start Step	End Step
1_baseline	0.0427	0.6274	0.6274	0.0427	↓0.5847	↓-93%	0	29
2_linear	0.0251	0.5738	0.5738	0.0251	↓0.5487	↓-96%	0	29
3_no_clip	0.0418	0.624	0.624	0.0452	↓0.5787	↓-93%	0	29
4_no_normalisation	0.0288	0.5815	0.5815	0.0319	↓0.5495	↓-95%	0	29
5_100_steps	0.0404	0.5971	0.5971	0.0404	↓0.5567	↓-93%	0	29
6_10_steps	0.0325	0.5937	0.5937	0.0351	↓0.5586	↓-94%	0	29

Lowering **sampling steps** saw an increase of convergence upfront with stagnation at higher epochs, especially at very low counts like 10. The phenomenon corresponds with the model producing learnt features very early on. Generated digits nevertheless converged to be more uniform and rather difficult to discriminate.

Conditional Generation

We first generated token embedding for labels similarly to timestep embedding, they share the same dimension and are added together in Unet. Note that the label is constantly checked for null value to polymorphically switch between unconditional and conditional generation.

```
class Unet(nn.Module):
    def __init__(...): # ...
        self.time_embedding=nn.Embedding(timesteps,time_embedding_dim)
        self.label_embedding=nn.Embedding(10,label_embedding_dim)
    def forward(self,x,t, label=None):
        t = self.time_embedding(t)
        if label is not None: t += self.label_embedding(label)
```

In the diffusion model class, label information is added to reverse diffusion via sampling along with timestep.

```
def _reverse_diffusion_with_clip(self, x_t: Tensor, t: Tensor, label, noise: Tensor) -> Tensor: #...
    if label is not None: label = torch.tensor([label] * n_samples).to(device)
    for i in range(self.timesteps - 1, -1, -1):
        x_t = self._reverse_diffusion_with_clip(x_t, t, label, noise)
def _reverse_diffusion_with_clip(self, x_t: Tensor, t: Tensor, label, noise: Tensor) -> Tensor:
    pred=self.model(x_t,t,label)
```

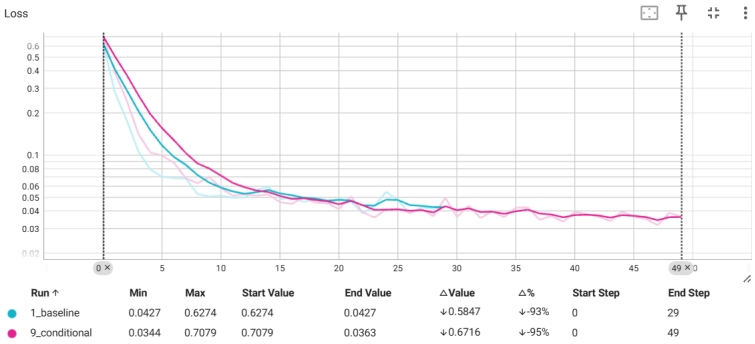
In the training script, MNIST labels are passed along the diffusion class into UNet. They are unused if no target label is provided.

```
for image, label in training_progress: # ...
    label = label.to(device) if args.label is not None else None
    pred = model(image, noise, label)
```

Finally, the I/O is extended to receive the label. Unconditional generation is used by default.

```
    parser.add_argument('--label',type=int, help='label for conditional generation', default=None)
# ...
samples=model_ema.module.sampling(args.n_samples, args.label, device=device)
```

To test our implementation, we made some customisation regarding input handling to the above implementation to generate all digits from 0 to 9 in order. Using default parameters for the experiments above, the conditional model achieved satisfactory results by the 40th epoch and continued to make minor incremental improvement after. It yields lower convergence than the unconditional model at lower epochs but reaches par soon after, likely due to the additional loss from combined embeddings.



Run	Min	Max	Start Value	End Value	△Value	△%	Start Step	End Step
1_baseline	0.0427	0.6274	0.6274	0.0427	↓0.5847	↓-93%	0	29
9_conditional	0.0344	0.7079	0.7079	0.0363	↓0.6716	↓-95%	0	49

Conditional generation of digits 0-9 at every 10th epoch up to the 50th.