Unconditional Diffusion Model on CIFAR-100

Ravindra Bisram and Danny Hong

As outlined in 'Diffusion models beat GANs on Image Synthesis' (Open AI), diffusion models are much better at image synthesis by being more faithful to the image. GANs have to produce an image in one go and generally have no options for refinement during generation. Diffusion is a slow and iterative process in which noise is converted into image, step by step. This allows diffusion models to have better options for guiding the image towards the desired result. For our midterm project, we built an unconditional diffusion model for both the MNIST and CIFAR-100 datasets, as well as a conditional diffusion model for the MNIST dataset. Our work is heavily inspired by 'Denoising Diffusion Probabilistic Models' (Ho et. al), which serves as a foundational paper for research into diffusion models.

The diffusion model is divided up into two steps: the forward noising process and the backward denoising process (which is a neural network). For the forward noising process, we created a Markov Chain with a fixed number of timesteps, where each step can be represented by $q(x_t \mid x_{t-1})$. The chain will take an input image and add small amounts of gaussian noise based on a linear variance schedule (learnable mean but fixed variance). Following the recommendations given in the DDPM paper, we chose to use 1,000 timesteps and a linear beta scheduler ranging from 0.0001 to 0.2. To avoid the slow process of iterating through the entire chain for a timestep, the paper reparamaterizes beta with a new variable alpha, which is 1 minus beta. This switch allows us to jump from the original image to any timestep. During the backward denoising process, we train a neural network to undo this process by removing small amounts of noise at every timestep. This process is represented by $p(x_{t-1} \mid x_t)$.

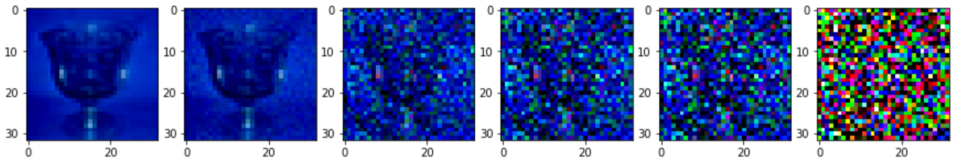
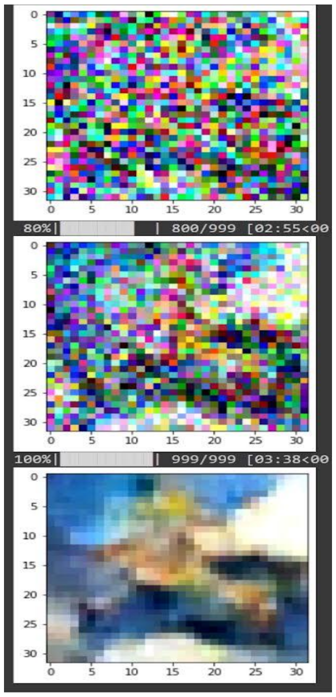


Figure 1: Forward noising process for a sample image for timesteps T = [0, 10, 100, 150, 195, 999]

The model used to learn the denoising process was a U-Net style image generating model. U-Net is a popular semantic segmentation architecture which progressively downsamples an input image followed by upsampling. The model also adds skip connections between layers that have the same resolution. This setup helps with the gradient flow during training, thus avoiding the vanishing gradient problem. The U-Net model takes in an input of a noisy image along with the corresponding timestamp in order to predict the noise that was added to the image at that given timestep. Our U-Net architecture consists of ResNet layers instead of traditional convolutional layers. Lastly, we've added attention layers into our model along with certain mechanisms to deal with timestamp encodings. In particular, the time encodings were passed into a Multi-Layer Perceptron module within the ResNet block. The Multi-Layer Perceptron converts the fixed sized time encodings into a vector that is compatible with the block dimensions in the ResNet layer. At this stage of the project we ran into a lot of shape errors as the data flowed through the upsampling and downsampling stages, which was resolved by painstakingly examining the output shapes produced by each layer and modifying the number of blocks in each stage.



```
80%|          | 800/999 [02:55<00
100%|          | 999/999 [03:38<00
```

Figure 2: The final iterations of the denoising process taking place for a sample image

For training specifications, we first preprocessed our data by normalizing the CIFAR100 data after loading it in as a tensorflow dataset. We used a batch size of 64 and trained the model for a total of 13 epochs. Lastly, for our optimizer, we decided to use Adam with a learning rate of 0.0001. The way we structured our model made it trivial to switch between the MNIST and CIFAR100 datasets by changing the number of channels. The loss after training the unconditional diffusion model on the CIFAR100 images after 13 epochs was 0.0387. As can be seen in Figure 3, the denoised images still look quite blurry. To improve the model's performance, we could train the model for more epochs. To save time we trained the model for a few epochs at a time and saved the weights so it could improve on its previous run each time. We could also look to implement a learning rate scheduler in which we vary the learning rate for each epoch. Nonetheless, we were able to successfully implement a neural network that can denoise images with additive Gaussian noise.

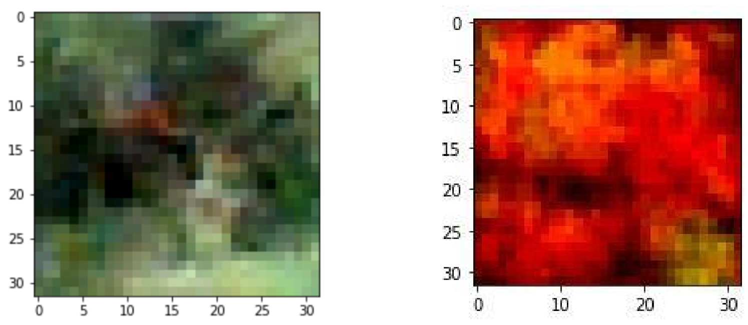


Figure 3: Sample generated output images after denoising from the unconditional diffusion model. Dog (left) and roses (right).

Once we had our model working for the unconditional case, we decided to try and implement a conditional model for the MNIST dataset as well. To achieve this, we encoded the class labels through embedding layers. Once we had the class label as a class embedding, we passed it through a dense layer followed by a reshaping step. This full "class conditioning" block was then concatenated with the input image at the beginning of the resnet block. The rest of the model could be left essentially the same. We tried to use this class conditioning block at the upsample, middle, and downsample stages, but ended up only needing to use them at the upsample and downsample blocks to get great results.

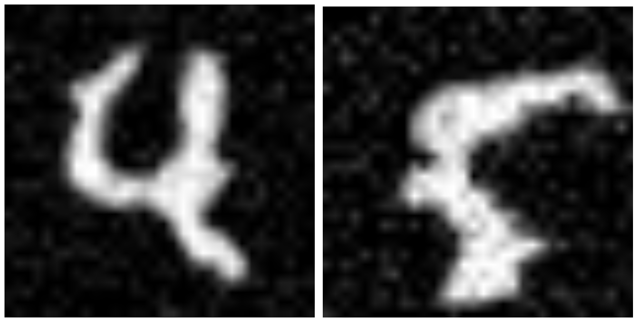


Figure 4: Conditional diffusion model outputs for MNIST dataset - class label 4 (left) and class label 5 (right)

Areas of potential future work on our project would include improving our model based on the newer DDIM paper (Denoising Diffusion Implicit Models), which uses a non-Markovian noising process which makes the reverse process much faster to sample from. Other research in the area also makes use of a cosine variance scheduler rather than linear, with promising results. We will also apply the conditional model on CIFAR100, it was only a matter of the time constraint that we did not do so already.

References

Class Conditioned Diffusion Model - https://medium.com/@vedantjumle/class-conditioned-diffusion-models-using-keras-and-tensorflow-9997fa6d958c#74cc

Denoising diffusion implicit models - https://arxiv.org/abs/2010.02502

Denoising Diffusion Implicit Models (Keras example) - https://keras.io/examples/generative/ddim/

Denoising diffusion probabilistic models - https://arxiv.org/abs/2006.11239

Diffusion Models as a Kind of VAE - https://angusturner.github.io/generative_models/2021/06/29/diffusion-probabilistic-models-I.html

Diffusion models beat GANs on Image Synthesis - https://arxiv.org/abs/2105.05233

Diffusion Models Made Easy - https://towardsdatascience.com/diffusion-models-made-easy-8414298ce4da

Diffusion Models | Paper Explanation | Math Explained - https://www.youtube.com/watch?v=HoKDTa5jHvg

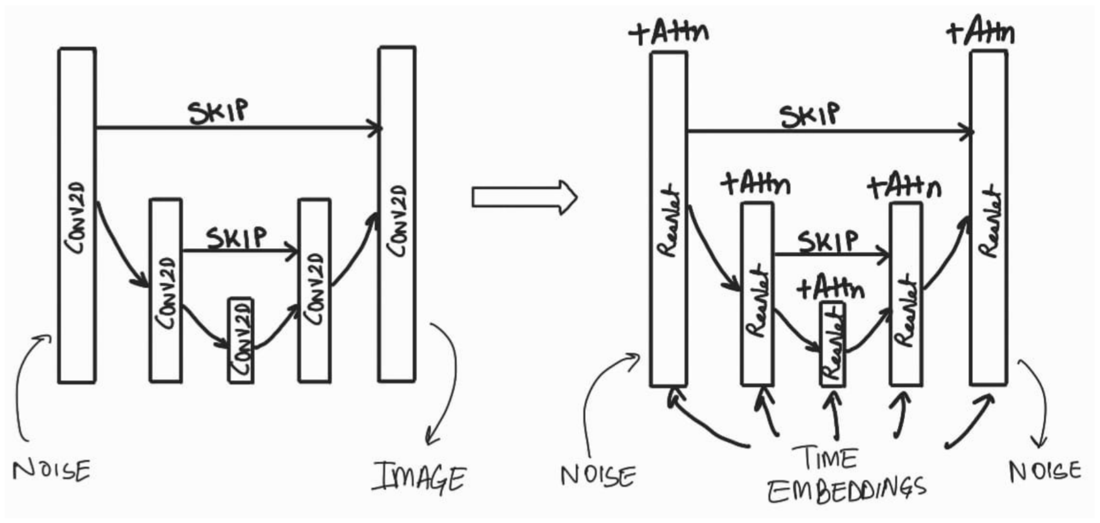
How diffusion models work: the math from scratch: https://theaisummer.com/diffusion-models/

Image generation with diffusion models using Keras and TensorFlow - https://medium.com/@vedantjumle/image-generation-with-diffusion-models-using-keras-and-tensorflow-9f60aae72ac

Introduction to Diffusion Models for Machine Learning - https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction/

Training a Conditional Diffusion Model from Scratch - https://wandb.ai/capecape/train_sd/reports/Training-a-Conditional-Diffusion-model-from-scratch--VmlldzoyNzIzNTQ1

Transformer Architecture: The Positional Encoding - https://kazemnejad.com/blog/transformer_architecture_positional_encoding/


SKIP
CONV2D
CONV2D
SKIP
CONV2D
CONV2D
CONV2D
NOISE
IMAGE
+Attn
+Attn
SKIP
+Attn
+Attn
ResNet
ResNet
SKIP
+Attn
ResNet
ResNet
ResNet
NOISE
TIME EMBEDDINGS
NOISE