Jazmine Brown

MD Report

ITAI 2376

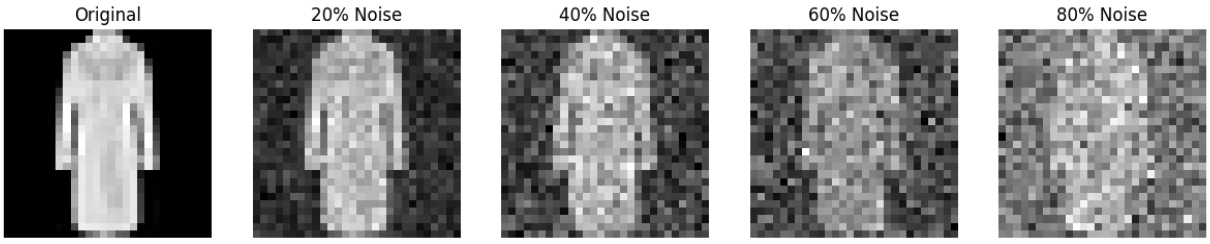
Patricia McManus

Understanding Diffusion

Explain what happens during the forward diffusion process, using your own words and referencing the visualization examples from your notebook.

During the forward diffusion process, a clean image is gradually corrupted by the incremental addition of Gaussian noise. At each of the 100-time steps, a small, pre-defined amount of noise controlled by the beta schedule is added to the image. Over time, these small perturbations accumulate until the image becomes nearly pure noise. Mathematically here is how progressive noise addition is represented. We start with a clean image, then add a little noise to the image step by step and after many steps the image becomes unrecognizable.

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon$$



For example, when you visualize the progression: At early steps, around 20% into the process, the image still retains much of its structure, though fine details may already be blurred. As the process continues around 40–60% progress, the essential structure starts to fade while noise dominates. Near the final steps above 80–90%, the image becomes almost indistinguishable from random noise.

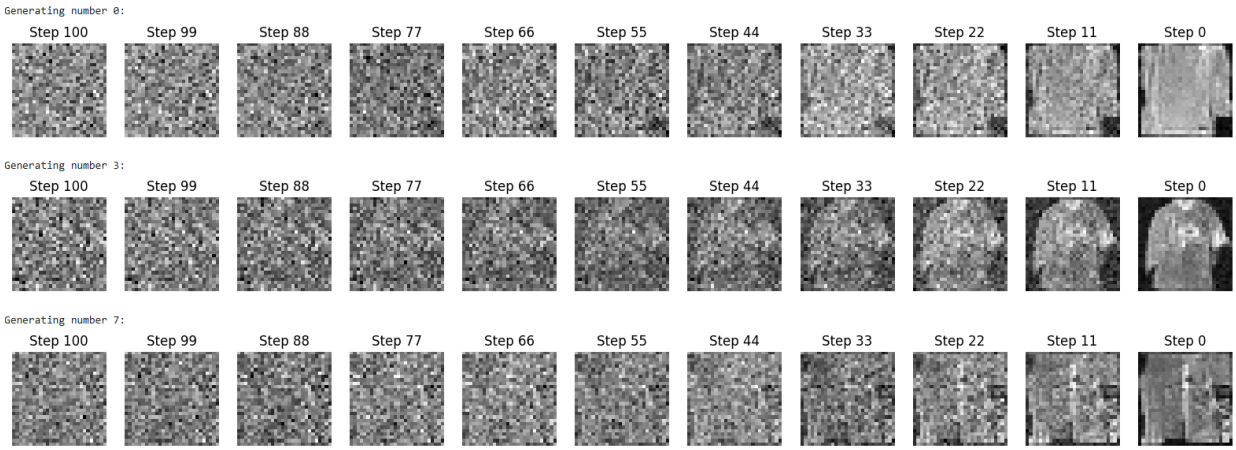
Why do we add noise gradually instead of all at once? How does this affect the learning process?

Adding noise gradually allows the model to learn the reverse process in small, manageable increments. By adding noise gradually, it creates many intermediate stages between a completely clean image and a fully noisy one. This smooth transition provides the learning algorithm with a range that is continuous of examples. This makes it easier for the model to learn the mapping in both directions forward and reverse diffusion.

If noise was added all at once, the jump from one image with high signal to one dominated by noise would be too abrupt, making the learning task extremely hard. If we added all noise at once the model would have no intermediate states to learn from, making it difficult to gradually denoise. With gradual diffusion, the model encounters a variety of noise levels, enabling it to handle different degrees of corruption.

Look at the step-by-step visualization - at what point (approximately what percentage through the denoising process) can you first recognize the image? Does this vary by image?

After examining the step-by-step visualization in the reverse process implemented in generate_samples and remove_noise, you start from noise t=99 t=99 t=99 and denoise to t=0 t=0 t=0. If you visualized this for example by plotting every 10 steps, you'd see the image emerge gradually. For Fashion-MNIST, recognizable shapes appear around 30-50% through denoising around t=50 t=50 t=50 to t=30 t=30 t=30, or 50%-70% of the 100 steps completed.



2. Model Architecture

Why is the U-Net architecture particularly well-suited for diffusion models? What advantages does it provide over simpler architectures?

The U-Net architecture is designed for tasks that benefit from both global context and fine spatial detail, which is essential in the denoising process of diffusion models. Down sampling path captures global context by progressively reducing spatial dimensions and increasing the number of feature channels. While up sampling path recovers spatial resolution and details, which is critical when reconstructing a clean image from noisy features. They connect corresponding layers in the encoder and decoder. These connections allow high-frequency fine-grained details to bypass the bottleneck, ensuring the final output retains necessary spatial details.

In our U-Net, after each down sampling block we save the intermediate feature maps as skip connections, which are later concatenated with up sampled features to help reconstruct clear images.

An advantage compared to simpler architectures like CNN, down sampling (28x28 >14x14>7x7) extracts high level features, while up sampling reconstructs details, crucial for denoising at various noise levels.

What are skip connections and why are they important? Explain them in relations to our model

Skip connections are direct links that connect corresponding layers in the encoder and decoder parts of the network, allowing high-resolution information to bypass several intermediate layers. In our model, the UNet.forward method collects the outputs from each downsampling block into a list these are the skip connections which are later concatenated with the upsampled features in the decoder. This design ensures that during the reconstruction process, the detailed spatial information lost during downsampling is reintroduced, leading to more accurate and detailed image generation.

These connections are particularly important because the downsampling process, while effective in extracting high-level abstract features, can reduce the spatial resolution significantly for example, from 28×28 to 7×7 and may eliminate fine details that are crucial for a clear reconstruction of the image. By reintegrating the high-resolution features through skip connections, the network can leverage both the abstract semantic information and the original image details to produce high-quality outputs.

Additionally, skip connections improve gradient flow during training. They provide shorter paths for the gradients to propagate back to earlier layers, which helps mitigate issues like vanishing gradients and contributes to overall training stability. For instance, the skip connection that passes features from an intermediate layer, such as one with dimensions [B, 64, 14, 14], into the corresponding decoder block ensures that important mid-level details edges and textures are preserved. This blending of preserved details

with the processed, upsampled features ultimately leads to a more robust and effective model for generating clear and recognizable images.

Describe in detail how our model is conditioned to generate specific images. How does the class conditioning mechanism work?

My model is conditioned on two key pieces of information: time conditioning and class conditioning.

Time conditioning is implemented through a time embedding mechanism. A sinusoidal positional embedding encodes the current time step of the diffusion process, effectively providing the model with a “timestamp” that indicates how much noise is present in the image at that stage. These embedding captures both absolute and relative time information. The time embedding is then processed by passing it through successive linear layers with GELU activations to enhance its representational power. Finally, it is projected to the proper dimensions and spatially reshaped if necessary, so that it can be added directly to the feature maps at the bottleneck of the network. This addition informs the model at each denoising step of its progress in the diffusion process, allowing it to adjust the denoising operations accordingly.

Class conditioning, on the other hand, is achieved by converting the class labels for example, representing clothing types like “T-shirt” or “Trouser” into a dense representation. The one-hot encoded class label is input into an embedding block composed of linear layers followed by a GELU activation. This block transforms the sparse one-hot vector into a rich embedding vector. After the transformation, the embedding is unflattened into a spatially broadcastable tensor with dimensions that match the intermediate feature maps in the network. Additionally, a conditioning mask is applied to allow for selective conditioning, which can be useful for approaches like classifier-free guidance. Finally, this class embedding is added to the intermediate feature maps in the network. This operation effectively steers the reverse diffusion process so that, as noise is progressively removed, the emerging image reflects the target clothing class.

Together, these conditioning mechanisms work in harmony to ensure that the reverse diffusion process generates images that are both clean and representative of the specified class. The time embedding guides the model regarding the noise level at each step, while the class embedding biases the generation toward producing images that match the desired clothing category. This dual conditioning results in a model capable of producing high-quality, class-specific images.

3. Training Analysis

What Does the Loss Value Tell Us?

The loss value in my diffusion model is a quantitative measure of how accurately the model predicts the noise that was initially added to the image during the forward diffusion process. In our implementation, we use Mean Squared Error (MSE) loss to compare the model's predicted noise with the actual noise, meaning that the loss directly reflects the discrepancy between these two values.

A high loss indicates that the model's predictions are far from the true noise values, suggesting that it has not yet learned an effective way to reverse the noise addition, whereas a low loss signifies that the predictions are very close to the actual noise, demonstrating that the model is successfully capturing the noise patterns necessary for reconstruction. As training progresses, the loss decreases from values above 1 in the early epochs to around 0.105–0.110 in later epochs indicating gradual improvement and convergence of the model's denoising capabilities. This steady decline in loss is closely correlated with the enhanced quality of generated images; clearer and more detailed reconstructions are produced as the model learns to estimate and remove the noise accurately.

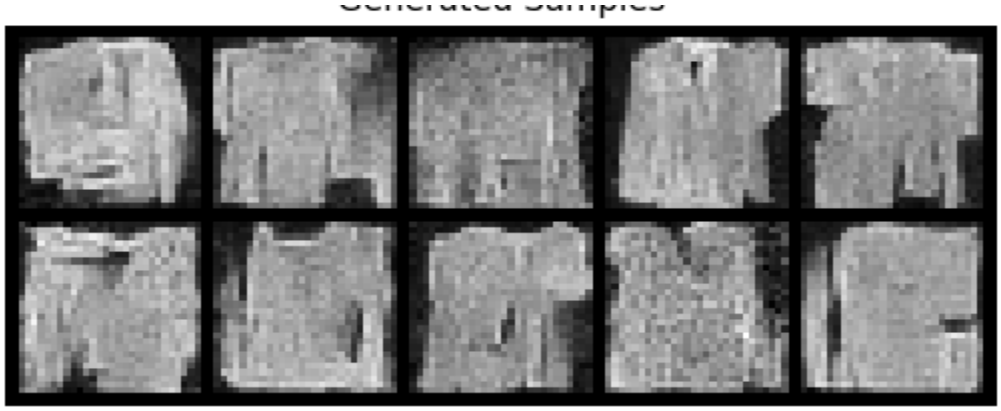
Moreover, the loss value serves as the primary signal for the optimizer, guiding weight updates throughout training, and is critical for diagnosing potential issues such as underfitting or overfitting. Monitoring this loss, therefore, not only informs us about the model's progress in learning the inverse diffusion process but also directly translates into higher quality, more reliable image outputs, which is particularly important for generating clear images of clothing items in the Fashion-MNIST dataset.

How did the quality of your generated images change throughout the training process?

Early in training, the generated images are very blurry and noisy, with little discernible structure. At this stage, the model has not yet learned to effectively remove the added noise, so the outputs bear little resemblance to recognizable clothing items.

As training progresses, you can observe a gradual improvement in image quality. Features begin to emerge, and the outlines and shapes of the clothing items become more defined. Although some noise may still be present, key characteristics such as the contours and textures of a T-shirt start to take shape, indicating that the model is beginning to capture the underlying structure.

By the end of the training process, the generated images become significantly sharper and convincingly resemble real clothing items from the Fashion-MNIST dataset. This improvement is evidenced not only by the visual progression in sample generations but also by the decreasing trend in validation loss over epochs. Both observations indicate that the model is effectively learning to reverse the diffusion process, reconstructing clear images from noisy inputs.



Why do we need the time embedding in diffusion models? How does it help the model understand where it is in the denoising process?

The time embedding in diffusion models plays a critical role by providing the model with an explicit indication of its progress through the diffusion process. It essentially tells the model “Where it is” in the sequence of noise addition or removal. This information is vital because the quantity and structure of the noise vary at different timesteps.

Knowing the specific timestep allows the model to adapt its denoising strategy accordingly. For instance, at early timesteps, when the noise is relatively low, the model needs to perform only minor corrections. In contrast, at later timesteps with higher noise levels it must remove more noise and reconstruct more of the original structure. Without the time embedding, the model would lack a crucial signal about the level of corruption in each image, which would hinder its ability to learn a consistent and effective denoising function across all noise levels.

4. CLIP Evaluation

What do the CLIP scores tell you about your generated images? Which images got the highest and lowest quality scores?

The CLIP scores provide a quantitative measure of how well the generated images match the intended textual descriptions, serving as an “automatic critic” for image quality. In our evaluation, three types of scores are computed by comparing the image embeddings to text prompts describing a “good” example, a “clear” version, and a “blurry” instance of a clothing item. High scores for the "good" or "clear" categories indicate that CLIP finds strong similarity between the generated image and the positive descriptive prompts, meaning the image is sharp, well-formed, and recognizable as the target clothing item. Conversely, a high score for the "blurry" category suggests that the image is of lower quality lacking clear details and suffering from excessive noise.

From my experiments, images that are produced later in the training process typically receive higher "good" and "clear" scores, demonstrating improved clarity and detail as the model learns to denoise more effectively. For instance, certain clothing items with simpler and more well-defined shapes such as a T-shirt often score highly, indicating successful reconstruction. On the other hand, images that still contain residual noise or lack sharpness possibly due to the inherent complexity or variability in the clothing item tend to receive higher "blurry" scores and are thus rated lower in overall quality.

Develop a hypothesis explaining why certain images might be easier or harder for the model to generate convincingly.

One hypothesis is that some clothing items in the Fashion-MNIST dataset are inherently easier for the model to generate because of their simpler, more consistent structure and limited variability. For example, items like T-shirts or trousers tend to have straightforward, well-defined shapes with clear edges and fewer intricate details. This simplicity makes it easier for the model to learn the reverse diffusion process and reconstruct the image from noise.

On the other hand, clothing items with more complex features such as sneakers, which might include subtle curves, textures, and varying shapes could be more challenging to generate convincingly. The increased complexity requires the model to capture and reproduce more detailed information during the denoising process. Additionally, if certain classes are better represented in the training dataset, the model has more examples to learn from, which may result in higher-quality generations for those items compared to classes with fewer examples or greater internal variability.

How could CLIP scores be used to improve the diffusion model's generation process? Propose a specific technique.

CLIP scores can be used to improve the diffusion model’s generation process by incorporating an additional guidance mechanism that evaluates the perceptual quality of the generated images in real time. One specific technique is to integrate a CLIP-based loss term into the training objective. In this approach, after the model generates an image, the CLIP model is used to compute the similarity between the generated image and target text descriptions for example "a clear, well-dressed T-shirt" for a particular clothing class. The loss term is then defined as the inverse of this similarity or as a penalty if the “blurry” or negative similarity is high, encouraging the diffusion model to produce images that score high on the desired attributes. By backpropagating this additional CLIP-guided loss alongside the original MSE loss, the model is explicitly incentivized to improve image clarity and adherence to the target class characteristics. This combined objective helps steer the denoising process not just toward removing

noise, but also toward generating images that are perceptually aligned with human-level expectations, ultimately resulting in higher quality and more recognizable outputs.

5. Practical Applications

How Could This Type of Model Be Useful in the Real World?

This type of diffusion model can be extremely useful in the real world in a variety of ways. For instance, in the fashion industry it could be used to generate synthetic images of clothing items, which is valuable for data augmentation to improve the performance of classification or recommendation systems.

Retailers might employ such models to create realistic images for virtual try-on systems or to preview new designs without the need for costly photoshoots. In addition, fashion designers could use these models as a creative tool to explore variations of styles or patterns, thus speeding up the design cycle.

Beyond fashion, diffusion models have broader applications in image restoration, creative art generation, and even medical imaging, where they help reconstruct high-quality images from degraded inputs. Overall, by generating detailed, high-quality images that are conditioned on specific attributes, these models can enhance both operational efficiency and innovation across multiple domains.

What Are the Limitations of Our Current Model?

The current diffusion model, while effective, has several limitations. First, it is computationally intensive both training and inference require many iterative denoising steps, which can lead to long processing times and high energy consumption. Additionally, the model demands significant GPU memory, particularly with high-resolution images or when using more complex architectures, which can limit its scalability and usability in resource-constrained environments.

The reverse diffusion process is also inherently stochastic, meaning that outputs can vary and may not always be consistent, which can make it challenging to reliably produce high-quality images. Furthermore, the current conditioning mechanisms are relatively simple and may not capture more nuanced style or contextual variations present in real-world fashion data. This can limit the model's ability to generate images that fully reflect the diverse range of clothing designs. Finally, while our evaluation with CLIP provides a helpful quality check, integrating such feedback into the training loop remains a challenge, as additional loss terms or guidance mechanisms can complicate optimization and require careful tuning to avoid destabilizing the learning process.

Bonus

If you were to continue developing this project, what three specific improvements would you make and why?

One improvement would be to modify the U-Net architecture by adding more layers or increasing the channel dimensions in the downsampling and upsampling blocks. This architectural enhancement could allow the network to capture more nuanced features and complex patterns present in fashion images. Although such changes are likely to increase the training time and memory usage due to a larger number of parameters, they can lead to higher-quality image generation through more detailed feature extraction and reconstruction.

A second improvement is to incorporate CLIP-guided selection into the generation process. Specifically, we could generate a larger pool of samples for instance, 15 per class and then use the CLIP model to evaluate each sample by comparing image features with text descriptions for example “a clear, well-defined T-shirt". By selecting the top three highest-quality samples based on the normalized similarity scores where high scores on positive prompts such as "good" and "clear" exceed those on negative prompts like "blurry", we can both improve the reliability of the final outputs and gather insights into what visual attributes CLIP considers high quality. This approach can serve as a post-processing step or be integrated into training as a supplementary loss term, though careful tuning would be required to avoid destabilizing the model.

Finally, enhancing the conditioning mechanism to support style conditioning would enable the generation of multiple styles for the same clothing item. For example, we could extend the class conditioning component to include style information such as descriptors for "slanted," "thick," or "thin" features either by concatenating an additional style embedding or by designing a multi-modal conditioning vector. Experimenting with this approach could help the model generate diverse visual variations for a given class, making the model more versatile and applicable for tasks like design variation or personalized fashion recommendations. Documenting these experiments would involve comparing image quality and diversity metrics across different styles, as well as analyzing how the extended conditioning affects convergence.