ZigMa: A DiT-style Zigzag Mamba Diffusion Model, Hebrew Review

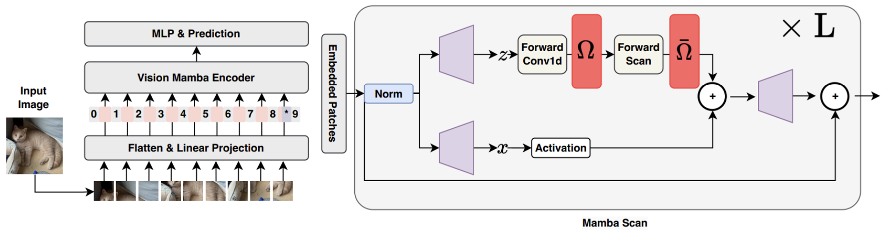


Figure 2: ZigMa. Our backbone is structured in L layers, mirroring the style of DiT [65]. We use the single-scan Mamba block as the primary reasoning module across different patches. To ensure the network is positionally aware, we've designed an arrange-rearrange scheme based on the single-scan Mamba. Different layers follow pairs of unique rearrange operation Ω and reverse rearrange $\bar{\Omega}$, optimizing the position-awareness of the method.

This paper caught my attention for a few reasons:

- There are diffusion models - my former love that I will soon reconnect with.
- There are State-Space Models (SSMs) here in the form of Mamba - my current passion that I'm just finishing up preparing a large presentation on and hopefully you'll see it soon as I am going to present it in meetups and conferences.
- The paper was published on April 1st and initially I was a bit suspicious 🙂

Additionally, the paper also has some of the Transformers (cross-attention) which adds to its completeness. Okay, so what do we have in this paper beyond several buzzwords? The paper proposes an interesting architecture intended for image and video generation. As mentioned, the architecture belongs to the family of generative diffusion models but contains components composed of SSMs (Mamba) in addition to the cross-attention core of Transformers. And there is an interesting innovation regarding the order in which image patches (or video frames) are inputted during training of the model.

Let's start with a brief explanation of generative diffusion models. Given a dataset (of images and/or videos) we train the network as follows:

- We add small amounts of Gaussian noise to a data sample until it becomes pure noise.
- We train a neural network (with Mamba with cross-attention in our case) to model the reverse process: i.e. from a noisy data sample at iteration n, to predict it at iteration n-1.

Once we have such a model we are actually able to generate an image from pure Gaussian noise in a gradual fashion, iteration by iteration. Over the years, many varied methods have emerged for how to add noise and exactly what the network should predict.

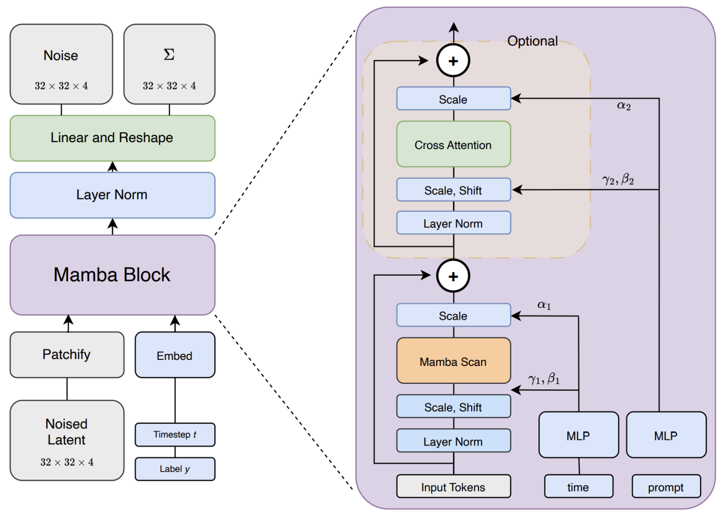
In the last 2 years there have been some interesting innovations in diffusion models, and since the paper uses them I have to tell you broadly what they are about (as mentioned I will talk about this in more depth in future reviews).

Recently, some interesting papers came out (e.g. https://arxiv.org/abs/2210.02747 and https://arxiv.org/abs/2303.08797 but there are dozens more) that generalize diffusion models to a continuous process of mapping a simple distribution (such as a standard Gaussian) to the data distribution (the complex distribution). This process is called continuous flow and its discretization (in the time domain, i.e. iterations) is a generative diffusion model for certain mappings. We have the freedom to choose the mapping (flow) between the data distribution and the simple distribution and there is quite a bit of research on how to choose it optimally (to maximize the quality of the generated data, stabilize the training process, make the mapping as simple or direct as possible, etc.).

So how does all this math relate to data generation? Well, there's a bit more math we'll need to dive into. Broadly speaking, the continuous flow between the simple distribution and the data distributions (sometimes called reverse-time) can be described by a stochastic differential equation (SDE) that contains:

- The noisy data itself x_t
- The velocity or rate of change (time derivative) of the flow at time t, v_t(x) (think of it as motion in the space between two clouds of points that can be defined by the directional velocity and starting or ending point).
- The score function s_t(x) which is essentially the log of p_t(x) - the distribution function of the noised data

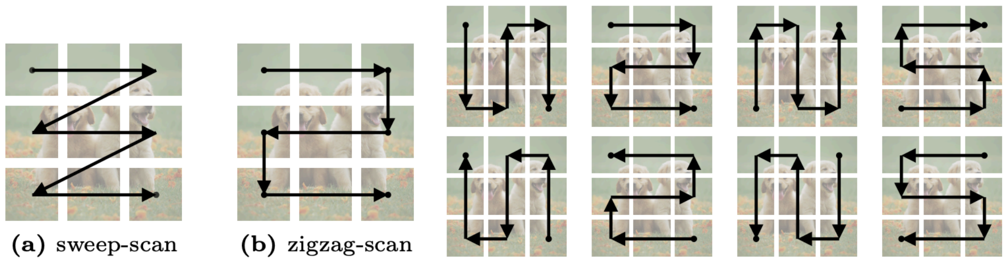
There is also a reverse-time Wiener process which constitutes the random (stochastic) part of this SDE.



So what can be done with this SDE, why is it needed? It turns out that we can formulate optimization problems that allow the estimation of s_t(x) and v_t(x) given training data. After estimating them, the SDE we talked about can be solved numerically (say by the Euler-Maruyama method) i.e. from a starting point sampled from the simple (Gaussian) distribution we can generate new data samples step-by-step. And this is exactly what is done in the paper.

Okay, we survived the math - now what's the connection to SSMs here? For that we need to recall the architecture of the Diffusion Transformer or DiT, which is the basis for OpenAI's SoRA engine. Essentially, DiT is composed of Transformer blocks whose purpose is to model the parameters s_t(x) and v_t(x) (of course after discretization in the time domain, i.e. iterations). The surveyed paper replaces the Transformer blocks with Mamba (and they also take the cross-attention which is the core of the Transformer but according to their figure this part is optional).

But here we have an issue. Since Mamba is an architecture intended for one-dimensional time-series sequences (like text tokens), here we have images which have two-dimensional connections between the patches (visual tokens) and 3D connections in videos (in addition we have frames). The paper adapts the SSM structure for the multi-dimensional input by combining multiple SSMs, where each one receives the input in a different order (see the figure below). That is, Mamba layers are stacked on top of one another and each input enters them in a different order (from what I understand, all Mambas work with the same parameter matrices A, B, C). This allows ZigMa to account for these connections. The paper extends this approach to video generation (for three-dimensional connections).


(a) sweep-scan (b) zigzag-scan (c) zigzag-scan with 8 schemes

I'll note that similar to DiT, the proposed (diffusion SSM-anchored) model operates in the latent space, meaning the input to the diffusion model is a latent representation of the data derived by the encoder. DiT uses a VAE encoder and decoder (one of its enhancements) but in this paper I couldn't understand whether the authors took a VAE or another encoder. In one place in the paper they hint that the encoder also contains an SSM but I couldn't find any further mentions of this.

The results look not bad, for around 2020 standards, but since this is one of the first papers combining SSMs and diffusion models we'll forgive them for that.

It turned out to be a long but hopefully more or less understandable review...