Myanmar Handwriting Generation

THAN LWIN AUNG

Handwriting Generation

- Handwriting Generation can be either “Online” or “Offline”.
- “Online” Handwriting Generation requires to capture Handwritten Strokes (Brush Points). Formally, Strokes (Brush Points) are defined as a Tuple **S (x, y, t)**, where **P(x, y)** defines the Position of a Brush Point, and T= **t** defines the Order of Brush Point.
- In some Literature, Strokes (Brush Points) are defined as a Tuple **S (x, y, e)** where **P(x, y)** defines the Position of a Brush Point, and E = **e** defines the Boolean Value to determine the End of a Stroke.
- However, “Offline” Handwriting Generation does not require to capture Strokes (Brush Points).

Strokes

$$S = < x, y, t >$$

Original Handwriting

Strokes

Stroke Generation

- Online Handwriting Generation with Deep Neural Network ultimately comes down to the Generation of Strokes, given a String of Glyphs.

- In other words, given **C** = $\{c_0, c_1, \ldots, c_n\}$ where **C** is a String Glyphs, what is the Probability of **S** = $\{s_0, s_1, \ldots, s_n\}$ where **S** = a Set of Strokes, each of which is defined by **s** = $\{<x_0, y_0, t_0>, <x_1, y_1, t_1>, \ldots, <x_m, y_m, t_m>\}$.

- Basically, Online Handwriting Generation can be assumed as a Sequence-to-Sequence Generation, which transduces a **String of Glyphs** to a **Sequence of Strokes**.

Stroke Generation with RNN

- In 2013, Graves[1] proposed one of the earliest methods of Online Handwriting Generation with a RNN based Transducer. In his method, he defined a Stroke as a Tuple $X_T = \langle x_1, x_2, x_3 \rangle$ where x_1 = Position X, x_2 = Position Y and x_3 = End of Stroke.

- Also, Graves[1] employed a Sliding Window with a **Scoring Function ϕ(t, u)** which defines the weighted score between the Sequence of Glyphs and the Sequence of Strokes. The weighted score also implicitly determines the **Alignment** between individual characters and individual strokes, especially when it comes to **Cursive Writing**.

$$\phi(t,u) = \sum_{k=1}^{K} \alpha_t^k \exp\left(-\beta_t^k \left(\kappa_t^k - u\right)^2\right)$$

$$w_t = \sum_{u=1}^{U} \phi(t,u) c_u$$

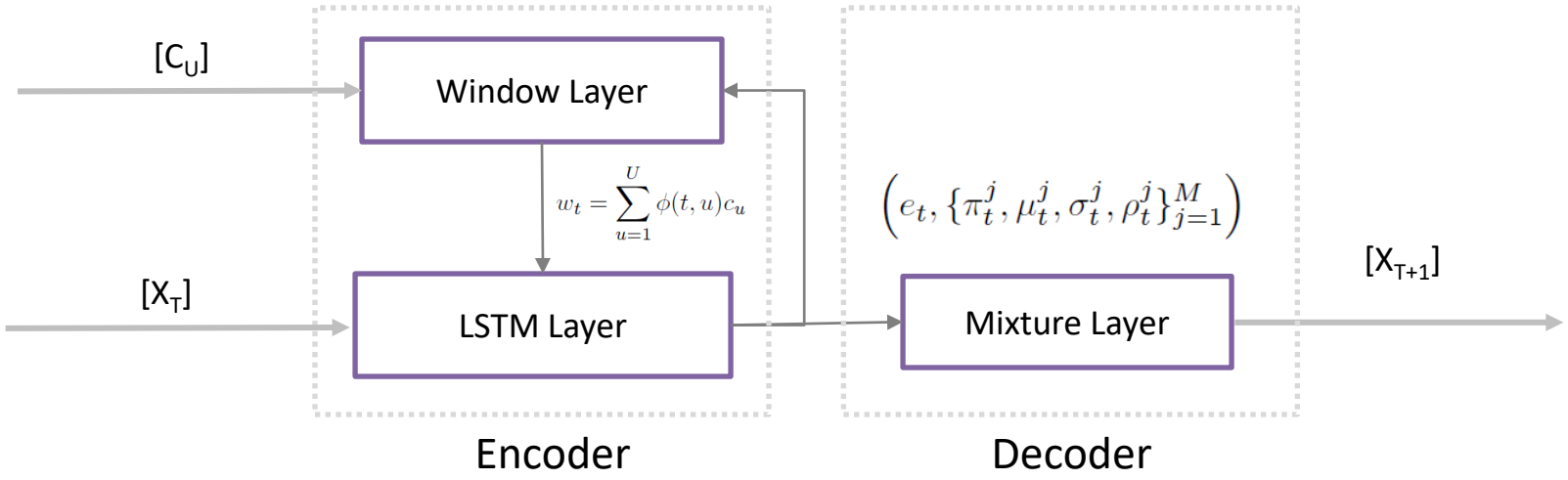
1. Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Next Stroke Generation

- For a Tuple $X_T = \langle x_1, x_2, x_3 \rangle$, a mixture of bivariate Gaussians was used to predict x_1 and x_2, while a Bernoulli distribution was used for x_3 , which indicates the End of Stroke. In short, RNN is optimized to learn the next Stroke Tuple, given previous Stroke Tuple along with the String of Glyphs.

$$\mathcal{L}(\mathbf{x}) = -\log \Pr(\mathbf{x}|\mathbf{c})$$

$$\Pr(\mathbf{x}|\mathbf{c}) = \prod_{t=1}^{T} \Pr\left(x_{t+1}|y_t\right)$$

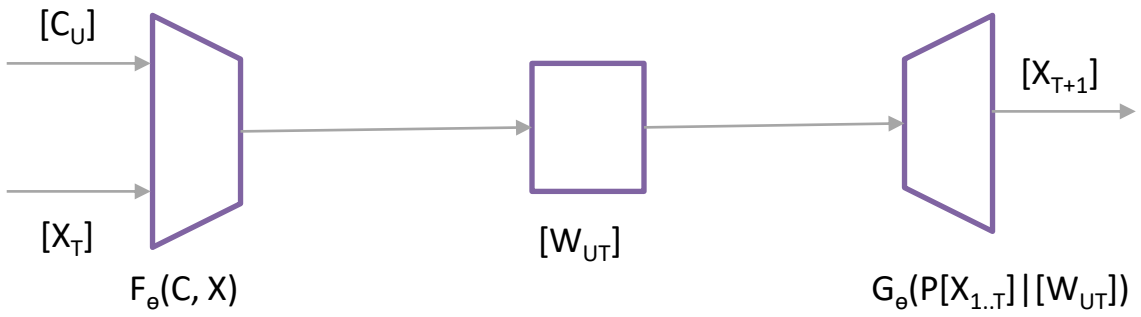


Basic RNN Transducer

Normally, RNN Transducer is defined as: Encoder F_θ, Latent Tensor $[W_{UT}]$ and Decoder G_θ.

$$[W_{UT}] = F_\theta([C_U], [X_T])$$

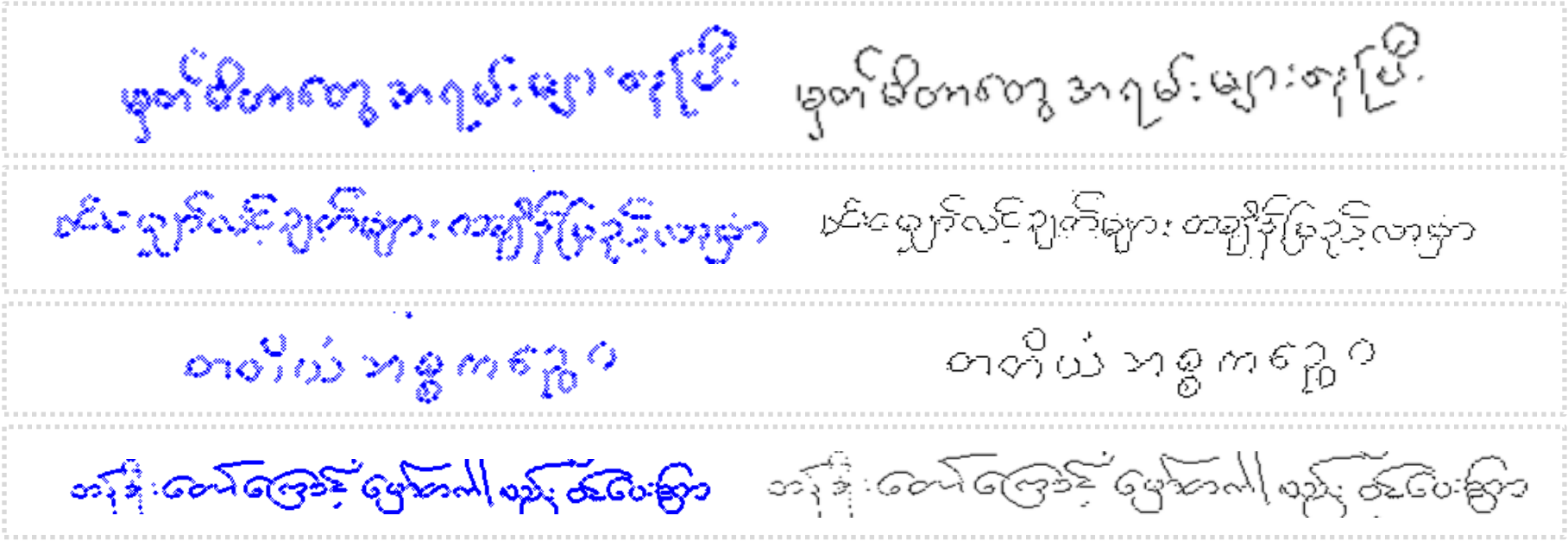
$$P[X_{t+1}] = G_\theta(P[X_{1..t}]|[W_{UT}])$$



Problems with Different Styles

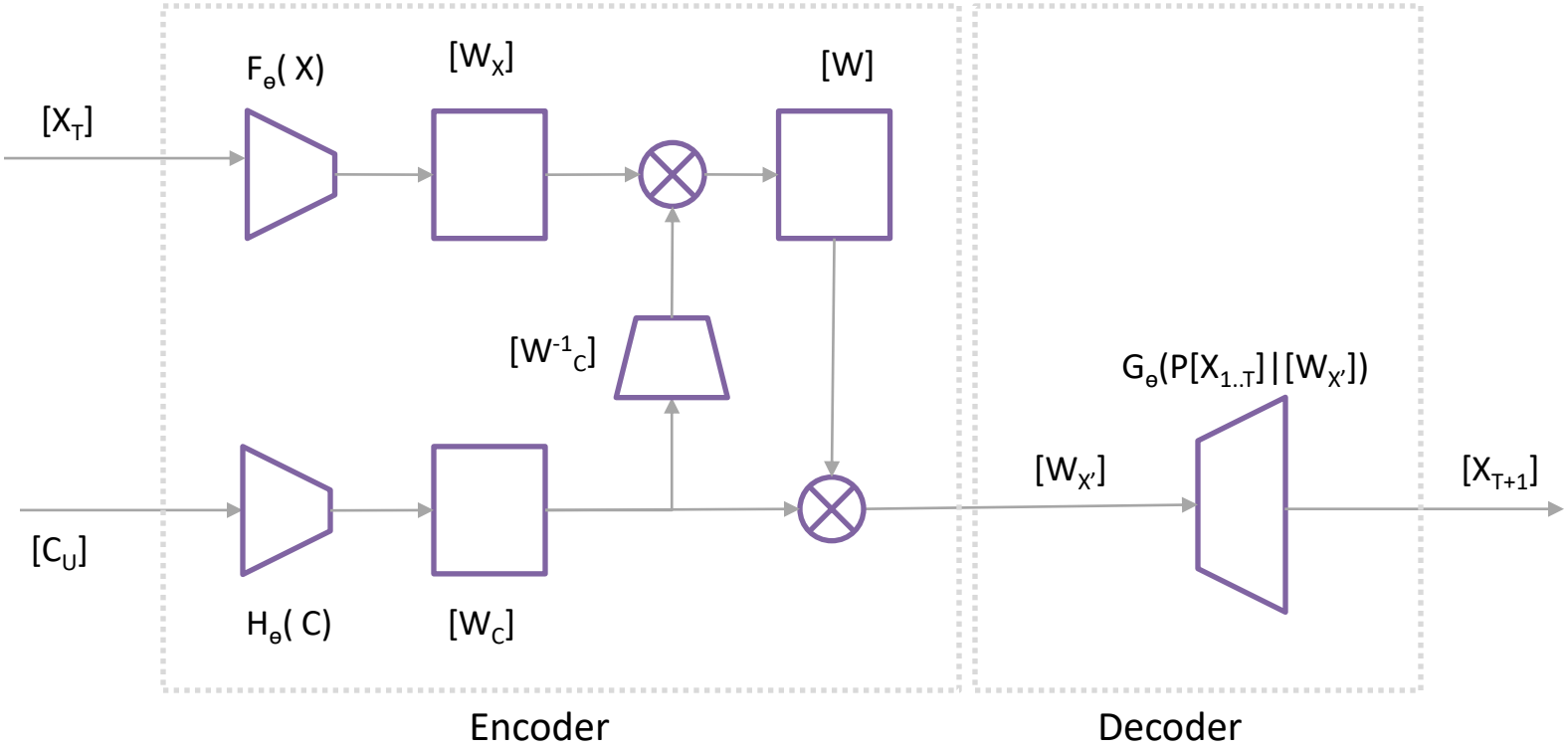
- Although RNN Transducer uses the Scoring Function $\phi(t, u)$, which jointly learns the alignments of the string of Glyphs and corresponding Strokes, the features are tightly coupled, which makes it harder to sample for different Writing Styles, as the **Latent Tensor** $[W_{UT}]$ is both **Styles and Glyphs Dependent**.

- Therefore, for Different Handwriting Styles, the DNN needs to separately learn the Strokes not only from the **String of Glyphs**, but also Styles from the **Set of Different Handwritings**.



Decoupling Styles and Glyphs

- In 2020, Kotani et. al[2] proposed Decoupled Style Descriptor (DSD) as to how to decouple the Styles and Glyphs for Online Handwriting Generation.



2. Kotani, A., Tellex, S., & Tompkin, J. (2020, August). Generating handwriting via decoupled style descriptors. In *European Conference on Computer Vision* (pp. 764-780). Cham: Springer International Publishing.

Decoupled Style Descriptor

- Kotani et. al[2] employed two independent Encoders for the Sequence of Glyphs (H_θ) and the Sequence of Strokes (F_θ).

- However, to define Decoupled Style Descriptor, additional Layers of Abstraction was implemented based on the following assumptions:

$$W_c = H_\theta(C_u)$$

$$W_x = F_\theta(X_t)$$

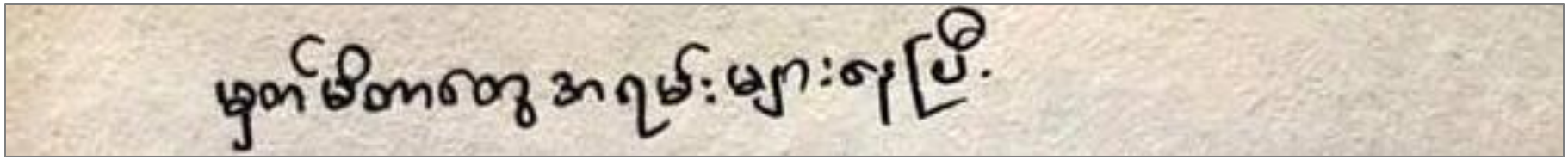
$$W_x = W_c \times W$$

$$W = W_c^{-1} \times W_x$$

, where **W = Decoupled Style Tensor**

Problems with Online Styles

- With Decoupled Style Descriptor, it has become easier to sample Different Writing Styles; however, the Writer Styles still need to be in Online (Strokes) Format during sampling.
- Acquiring Writer Styles as Strokes is quite challenging task as most Handwritten Texts are only available in Images, and extracting Stroke Points from Images are really challenging task.



Original Handwritten Image

Writer Style Strokes

Style Conditioning with Diffusion Model

- In 2020, Luhman et al.[3] proposed a Diffusion Model to eliminate the necessity of writer styles in online format during sampling.
- Luhman et al.[3] employed three inputs: the **String of Glyphs** [C_u], the **Sequence of Strokes** $S_T = <s_1, s_2, s_3>$, and finally the **Image Tensor** of Handwritten Image [X], during training.
- However, only two inputs: the **String of Glyphs** [C_u] and **Image Tensor** [X], are necessary during sampling and generation, thereby eliminating the requirement of Stroke Samples.

3. Luhman, T., & Luhman, E. (2020). Diffusion models for handwriting generation. *arXiv preprint arXiv:2011.06704*.

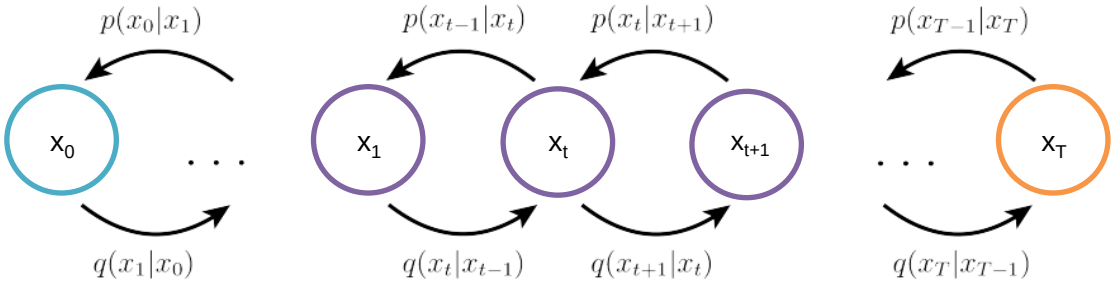
Variational Diffusion

- In Variational Diffusion, the distribution of each Latent Variable in Encoder is Gaussian (Noise) .

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})$$

- Therefore, the distribution of Encoder is: $\prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$

- For Decoder: $p(\boldsymbol{x}_T)\prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, where $p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \mathbf{0}, \mathbf{I})$



Denosing

- If each encoding step is actually just adding Gaussian Noise: $\mathbf{q(x_t \mid x_{t-1})}$

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathbf{I})$$

- Then, each decoding step is Reversing of Gaussian Noise or Denoising : $\mathbf{q(x_{t-1} \mid x_t)}$

- However, $\mathbf{q(x_{t-1}, x_t)}$ is not tractable. Therefore, $\mathbf{q(x_{t-1} \mid x_t, x_0)}$ is jointly conditioned on x_0 assuming how much noise is added from x_0 to x_{t-1}, given x_t.

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)q(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}$$

- Then, the distribution of Decoder: $\mathbf{p_\theta(x_{t-1} \mid x_t)}$ is actually to estimate the distribution of denoising: $\mathbf{q(x_{t-1} \mid x_t)}$.

Evidence Lower Bound

$$\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\right] - D_{\mathrm{KL}}(q(\boldsymbol{x}_T|\boldsymbol{x}_0) \parallel p(\boldsymbol{x}_T)) - \sum_{t=2}^{T} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]$$

- In 2022, Luo[4] suggested that Evidence Lower Bound (ELBO) is primarily used to approximate the Lower Bound Estimate of the Latent Variable Model.

- However, to estimate the distribution of denoising from ELBO:

$$\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]$$

- Therefore, Denosing in Diffusion Model ultimately comes down to estimating the KL Divergence between $\mathbf{q(x_{t-1} \mid x_t, x_0)}$ and $\mathbf{p_\theta(x_{t-1} \mid x_t)}$.

- Since $\mathbf{q(x_{t-1} \mid x_t, x_0)}$ is a Gaussian Distribution: $\dfrac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}$

- Then, $\mathbf{p_\theta(x_{t-1} \mid x_t)}$ is also a Gaussian Distribution: $\dfrac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)}{1-\bar{\alpha}_t}$

4. Luo, C. (2022). Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*.

Noise Estimation

- However, Diffusion Model mostly employs KL Divergence of Noise Estimation between Decoding and Encoding, instead.

$$\boldsymbol{x}_0 = \frac{\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0}{\sqrt{\bar{\alpha}_t}}$$

- Therefore, $\mathbf{q(x_{t-1} \mid x_t, x_0)}$ = μ_q (x_t, x_0) = $\frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}_0$

- Likewise, $\mathbf{p_\theta(x_{t-1} \mid x_t)}$ = μ_θ (x_t, t) = $\frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$

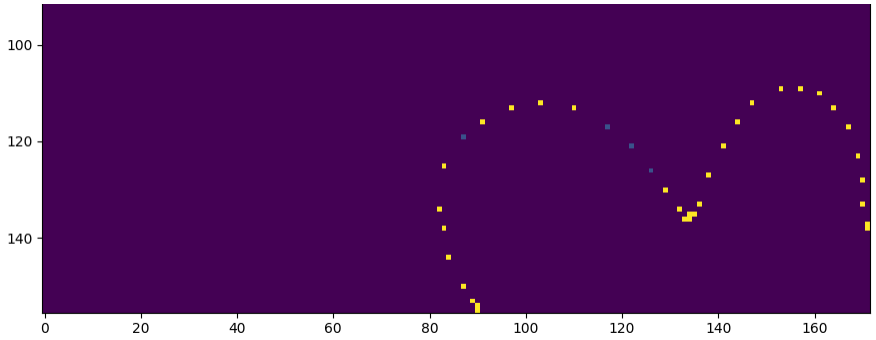
- Therefore, Noise Estimation becomes:

$$\arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t} \left[\|\boldsymbol{\epsilon}_0 - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\|_2^2 \right]$$

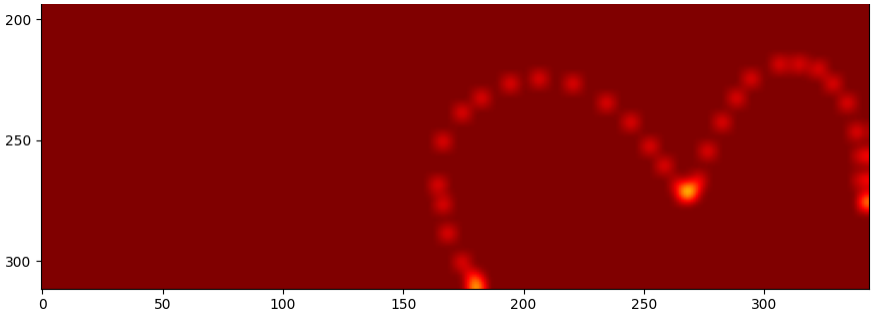
Stroke Labels

$$S = < x, y, t >$$

For each Glyph, corresponding Stroke Points are captured.



Points



Gaussian

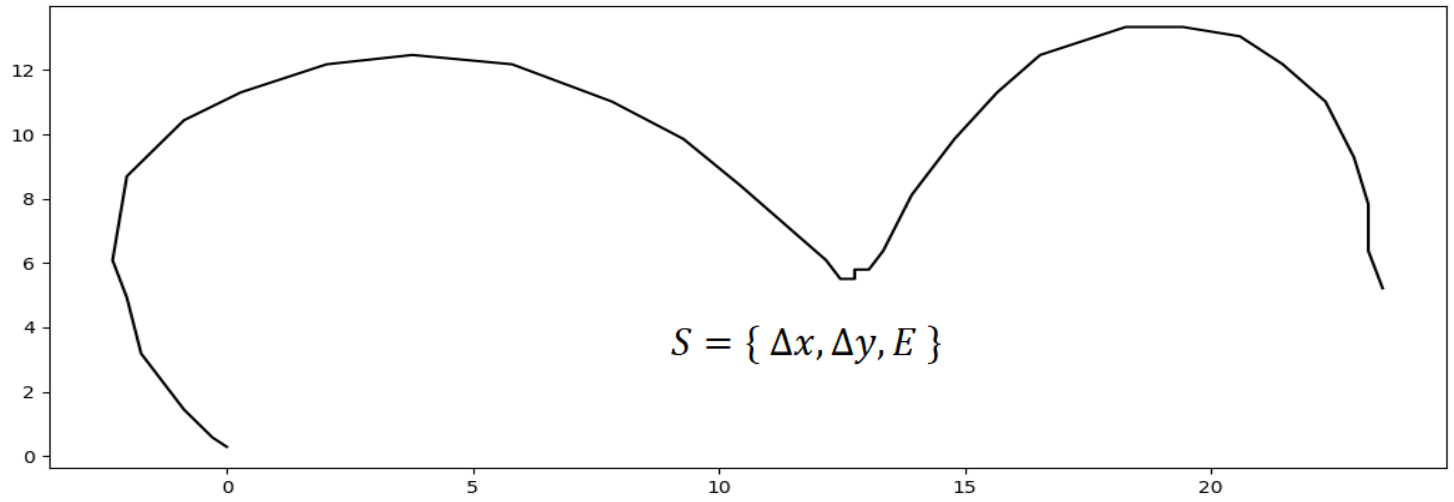
```
<Glyphs>
<Glyph  text="∞">
<Stroke>
<Point x="90.078125" y="155.00" time="0" />
<Point x="90.078125" y="154.00" time="1" />
<Point x="89.078125" y="153.00" time="2" />
<Point x="87.078125" y="150.00" time="3" />
<Point x="84.078125" y="144.00" time="4" />
<Point x="83.078125" y="138.00" time="5" />
<Point x="82.078125" y="134.00" time="6" />
<Point x="83.078125" y="125.00" time="7" />
<Point x="87.078125" y="119.00" time="8" />
<Point x="91.078125" y="116.00" time="9" />
<Point x="97.078125" y="113.00" time="10" />
<Point x="103.078125" y="112.00" time="11" />
<Point x="110.078125" y="113.00" time="12" />
<Point x="117.078125" y="117.00" time="13" />
<Point x="122.078125" y="121.00" time="14" />
<Point x="126.078125" y="126.00" time="15" />
<Point x="129.078125" y="130.00" time="16" />
<Point x="132.078125" y="134.00" time="17" />
<Point x="133.078125" y="136.00" time="18" />
<Point x="134.078125" y="136.00" time="19" />
<Point x="134.078125" y="135.00" time="20" />
<Point x="135.078125" y="135.00" time="21" />
<Point x="136.078125" y="133.00" time="22" />
<Point x="138.078125" y="127.00" time="23" />
<Point x="141.078125" y="121.00" time="24" />
<Point x="144.078125" y="116.00" time="25" />
<Point x="147.078125" y="112.00" time="26" />
<Point x="153.078125" y="109.00" time="27" />
<Point x="157.078125" y="109.00" time="28" />
<Point x="161.078125" y="110.00" time="29" />
<Point x="164.078125" y="113.00" time="30" />
<Point x="167.078125" y="117.00" time="31" />
<Point x="169.078125" y="123.00" time="32" />
<Point x="170.078125" y="128.00" time="33" />
<Point x="170.078125" y="133.00" time="34" />
<Point x="171.078125" y="137.00" time="35" />
<Point x="171.078125" y="138.00" time="36" />
</Stroke>
</Glyph>
```

Stroke Dataset

$$S = < x, y, t >$$

- Stroke Points are redefined as **ΔX** = offset from previous X, **ΔY** = offset from previous Y, and **E** = End of Stroke.

$$S = \{ \Delta x, \Delta y, E \}$$



Cost Functions

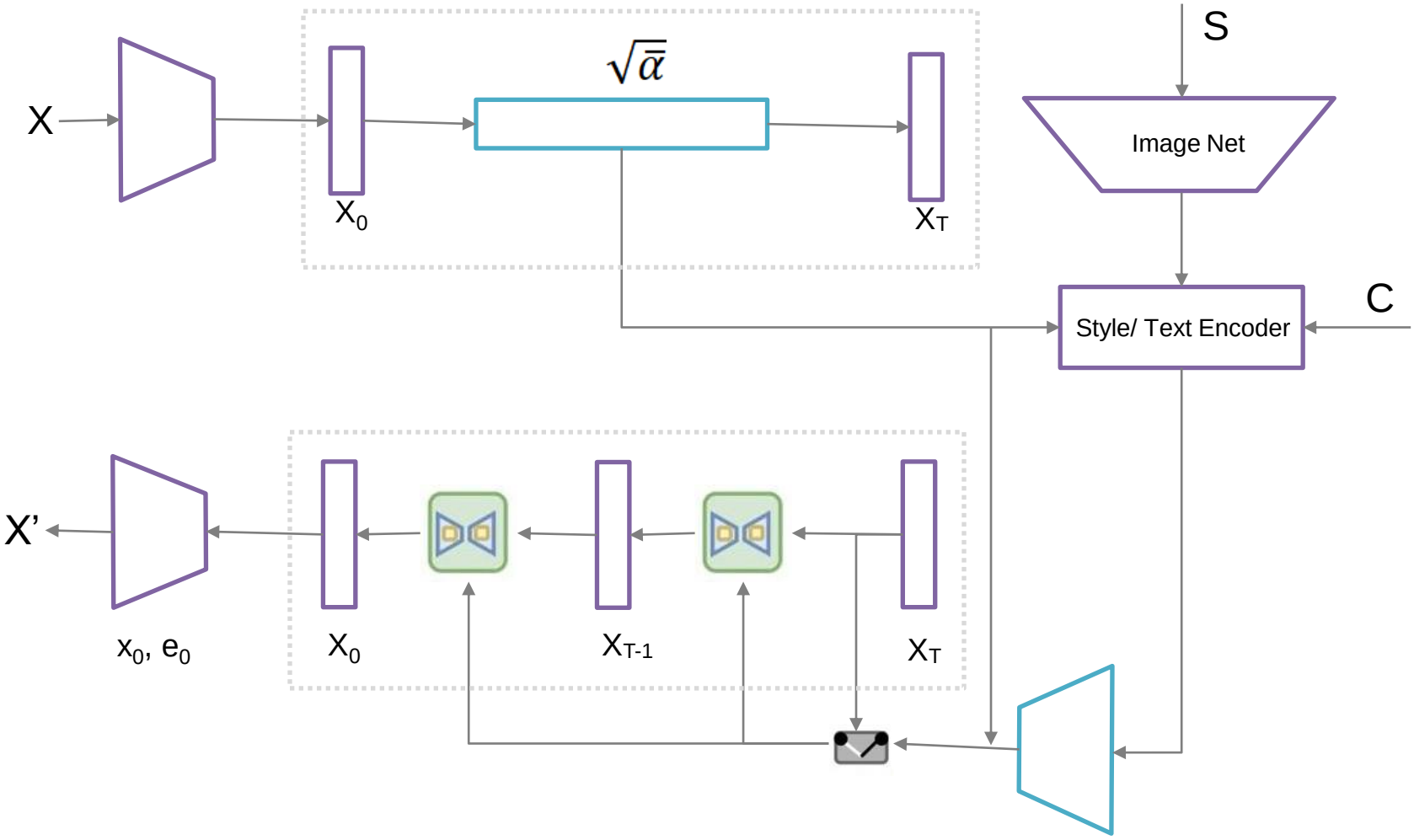
- With the redefinition of Strokes, the Noise Estimation finally has two parts: Estimating Stroke Offsets { **ΔX, ΔY** } and Estimating End of Strokes {**E**}.
- Therefore, accordingly, Luhman et al.[3] proposed two cost functions:

$$L_{stroke}(\theta) = \left\| \epsilon - \epsilon_\theta(x_t, c, s, \sqrt{\bar{\alpha}}) \right\|_2^2$$

$$L_{end}(\theta) = -e_0 log(e) - (1 - e_0)\log(1 - e)$$

3. Luhman, T., & Luhman, E. (2020). Diffusion models for handwriting generation. *arXiv preprint arXiv:2011.06704*.

Architecture



Training and Sampling

$$x_0 \sim q(x_0)$$

$$t \sim Uniform(\{1, \dots, T\})$$

$$\sqrt{\bar{\alpha}} \sim Uniform(l_{t-1}, l_t)$$

$$\epsilon \sim \mathrm{N}(0, \mathrm{I})$$

$$x_t = \sqrt{\bar{\alpha}} x_0 + \sqrt{1 - \bar{\alpha}}\epsilon$$

$$\nabla_\theta (L_{stroke}(\theta) + \bar{\alpha} L_{end}(\theta))$$

$$x_t \sim \mathrm{N}(0, \mathrm{I})$$

$$for\ t = T, \dots, 1\ do$$

$$z \sim \mathrm{N}(0, \mathrm{I})$$

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_0 - \sqrt{1 - \bar{\alpha}_t}\epsilon_\theta(x_t, c, s, \sqrt{\bar{\alpha}_t}))$$

$$x_{t-1} = x_{t-1} + \sqrt{1 - \bar{\alpha}_{t-1}} z$$

$$e_0 = e_\theta(x_t, c, s, \sqrt{\bar{\alpha}_t})$$

Results

The project is currently in progress. I will post the results as soon as possible.