UNIVERSITAS CLAUDIOPOLITANA • BABEŞ-BOLYAI •

TRADITIO NOSTRA
UNACUM EUROPAE
VIRTUTIBUS SPLENDET

1581
ROMÂNIA

UNIVERSITAS CLAUDIOPOLITANA

TRADITIO NOSTRA
UNACUM EUROPAE
VIRTUTIBUS SPLENDET

1581

ROMÂNIA

• BABEȘ-BOLYAI •

UNIVERSITAS CLAUDIOPOLITANA

TRADITIO NOSTRA
UNACUM EUROPAE
VIRTUTIBUS SPLENDET

1581
ROMÂNIA

• BABEŞ-BOLYAI •

```python
app.add_middleware(
    CORSMiddleware,
    allow_origins=["http://localhost:3000"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

```python
models_available = {
    "transformer": False,
    "vae": False,
    "markov": False
}
```

```
def parse_midi(midi_path):
    midi_data = pretty_midi.PrettyMIDI(midi_path)
    feature = np.zeros(128, dtype=np.float32)

    for instrument in midi_data.instruments:
        for note in instrument.notes:
            feature[note.pitch] += note.velocity / 127.0

    if np.sum(feature) > 0:
        feature = feature / np.sum(feature)
    return feature
```

NOTE_ONNOTE_OFFCONTROL_CHANGE

NOTE_ON_60TIME_SHIFT_4VELOCITY_80TIME_SHIFTDURATION

TIME_DELTA

$NP(s_t)\exp\left(-\frac{1}{N}\sum_t \log P(s_t)\right)$

$p_i i - \sum_i p_i \log p_i$

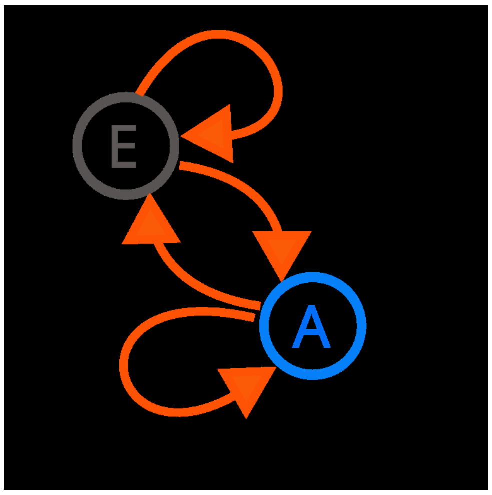
`hmmlearn`

$$P(X_{t+1} = s_j \mid X_t = s_i, X_{t-1} = s_k, \dots) = P(X_{t+1} = s_j \mid X_t = s_i) = p_{ij}$$

$\mathcal{S} = \{s_1, s_2, \dots, s_N\}$

$\mathbf{P} = [p_{ij}] p_{ij} \geq 0 \sum_j p_{ij} = 1 i$

$p_{ij} s_i s_j$



EA

nn

$$P(X_{t+1} \mid X_t, X_{t-1}, \ldots, X_{t-n+1})$$

$\mathcal{O}(|\mathcal{S}|^n)$

$\mathcal{S}$

p_{ij}

$$p_{ij} = \frac{(s_i \to s_j)}{\sum_k (s_i \to s_k)}$$

$\pi\pi$

$$\pi_j = \sum_i \pi_i p_{ij} \forall j$$

π

H

$$H = -\sum_{i,j} \pi_i p_{ij} \log p_{ij}$$

- $*$
- $*$
- $*$

$$= |\mathcal{S}|^{n+1}$$

$|\mathcal{S}| = 50 n = 3$

$\mathbf{P}$

$\mathbf{P} s_0 T [s_0, s_1, \ldots, s_T] t = 1 T s_t \sim (\mathbf{P}[s_{t-1}, :])$

$$\hat{p}_{ij} = \frac{(s_i \to s_j) + \lambda}{\sum_k ((s_i \to s_k) + \lambda)} \frac{(s_i \to s_k) + \lambda)}{}$$

$|\mathcal{S}| = 65$

$H = 2, 3$

`MarkovChain`

`hmmlearnCuPymusic21sklearncuML`

$\mu, \sigma,$

μ, σ

μ

$\rightarrow_n otemappingekettartfenn. Ezamegkzeltslehetvtesziazeneifrzisoksmotvumokpontosabbmodellezst,$

$$P(X_{t+1}|X_{t-n+1}, X_{t-n+2}, \ldots, X_t) = \frac{(X_{t-n+1}, \ldots, X_t, X_{t+1})}{(X_{t-n+1}, \ldots, X_t)}$$

$${}_{t+1} = {}_{t+1} - {}_{t}$$

$$P(X_{t+1} = j | X_t = i) = \frac{(i \to j)}{\sum_k (i \to k)}$$

$$P'(X_{t+1} = j | X_t = i,) = \begin{cases} P(X_{t+1} = j | X_t = i) j \in () \\ 0 \end{cases}$$

$$P(\mathbf{O}|\lambda) = \sum_{\mathbf{Q}} P(\mathbf{O}|\mathbf{Q}, \lambda) P(\mathbf{Q}|\lambda)$$

$$P(O_t | q_t = j) = \mathcal{N}(O_t; \mu_j, \Sigma_j)$$

$\lambda = (A, B, \pi)$

$$\begin{aligned} A &= \{a_{ij}\} \\ B &= \{b_j(o_k)\} \\ \pi &= \{\pi_i\} \end{aligned}$$

$\lambda O = O_1, O_2, \ldots, O_T P(O|\lambda)$

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_t, q_t = S_i|\lambda)$$

$$\begin{aligned} \alpha_1(i) &= \pi_i b_i(O_1) \\ \alpha_{t+1}(j) &= \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}) \\ P(O|\lambda) &= \sum_{i=1}^{N} \alpha_T(i) \end{aligned}$$

$$Q^* = \arg\max_{Q} P(Q|O,\lambda)$$

$$\delta_t(i) = \max_{q_1,\ldots,q_{t-1}} P(q_1,\ldots,q_{t-1},q_t=i,O_1,\ldots,O_t|\lambda)$$
$$\psi_t(j) = \arg\max_{1\le i\le N}[\delta_{t-1}(i)a_{ij}]$$

$$\gamma_t(i) = P(q_t=S_i|O,\lambda) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N}\alpha_t(j)\beta_t(j)}$$
$$\xi_t(i,j) = P(q_t=S_i,q_{t+1}=S_j|O,\lambda)$$

$$\pi_i^{(j)} = \gamma_1(i)$$
$$a_{ij}^{(j)} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)}$$
$$b_j(k)^{(j)} = \frac{\sum_{t=1,O_t=v_k}^{T}\gamma_t(j)}{\sum_{t=1}^{T}\gamma_t(j)}$$

$$b_j(O_t) = \mathcal{N}(O_t;\mu_j,\Sigma_j) = \frac{1}{\sqrt{(2\pi)^d|\Sigma_j|}}\exp\left(-\frac{1}{2}(O_t-\mu_j)^T\Sigma_j^{-1}(O_t-\mu_j)\right)$$

$\mu_j \Sigma_j j$

$$P(X_{t+1}|X_t^{(n)}, H_t) = \sum_h P(X_{t+1}|X_t^{(n)}, H_t = h)P(H_t = h|X_t^{(n)})$$

$X_t^{(n)} = (X_{t-n+1}, \dots, X_t)n$

H_t

$P(H_t = h|X_t^{(n)})$

$$P(_{t+1}|_t^{(n)}, _t, _t)$$

`hmmlearn`

`sklearn`

cuML

```
FUNCTION initialize_hmm(sequences):
    features = extract_musical_features(sequences)
    clusters = kmeans_clustering(features, n_hidden_states)
    hmm_model = GaussianHMM(n_components=n_hidden_states)
    hmm_model.fit(features)
    RETURN hmm_model

FUNCTION generate_sequence(length):
    IF hmm_model exists:
        states, observations = hmm_model.sample(length)
        RETURN states
    ELSE:
        RETURN fallback_sequence(length)
```

$$y = \sigma\big(W_2\,\sigma(W_1 x + b_1) + b_2\big),$$

$W_1, W_2 b_1, b_2 \sigma$

$t h_t x_t h_{t-1}$

$$h_t = \sigma\big(W_h h_{t-1} + W_x x_t + b\big),$$

$W_h W_x b \sigma$

$W_h \sigma W_h$

$h_t c_t$

$$\begin{aligned}
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f), \\
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i), \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o), \\
\tilde{c}_t &= \tanh(W_c[h_{t-1}, x_t] + b_c), \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \\
h_t &= o_t \odot \tanh(c_t).
\end{aligned}$$

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

$p(\mathbf{z})$

$p_\theta(\mathbf{x}|\mathbf{z})$

$\mathbf{x}\mathbf{z}$

$p_\theta(\mathbf{z}|\mathbf{x})q_\phi(\mathbf{z}|\mathbf{x})$

$$\log p_\theta(\mathbf{x}) \geq \underbrace{\mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x}|\mathbf{z})]}_{} - \underbrace{D(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}))}_{}$$

$$\overrightarrow{\mathbf{h}}_t = (\mathbf{x}_t, \overrightarrow{\mathbf{h}}_{t-1})$$
$$\overleftarrow{\mathbf{h}}_t = (\mathbf{x}_t, \overleftarrow{\mathbf{h}}_{t+1})$$
$$[\boldsymbol{\mu}, \boldsymbol{\sigma}] = ([\overrightarrow{\mathbf{h}}_T; \overleftarrow{\mathbf{h}}_1])$$

$$p_\theta(x_t|\mathbf{z}, x_{<t}) = (\mathbf{z}, x_{t-1}, \mathbf{h}_{t-1})$$

$$\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$$
$$\mathbf{z} \sim p(\mathbf{z}|\mathbf{z})$$

$$\mathbf{z} = \underbrace{\boldsymbol{\mu}} + \boldsymbol{\sigma} \odot \underbrace{\boldsymbol{\epsilon}}_{\boldsymbol{\epsilon} \sim \mathcal{N}(0,\mathbf{I})}$$

$\boldsymbol{\mu}\boldsymbol{\sigma}\boldsymbol{\epsilon}$

$$\mathcal{L}_\beta = \mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \beta \cdot D(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}))$$

$\beta > 1$

$$D = -\frac{1}{2} \sum_{j=1}^{J} \left(1 + \log \sigma_j^2 - \mu_j^2 - \sigma_j^2\right)$$

J

μ_j^2

$\log \sigma_j^2 - \sigma_j^2$

β

$$\mathcal{L} = \mathbb{E}_q[\log p(\mathbf{x}|\mathbf{z})] - \beta_t D$$

$$y_i = \frac{\exp((\log \pi_i + g_i)/\tau)}{\sum_j \exp((\log \pi_j + g_j)/\tau)}, g_i \sim (0, 1)$$

$$\mathbf{z}_q = \arg\min_{\mathbf{e}_k \in \mathcal{C}} \|\mathbf{z}_e - \mathbf{e}_k\|_2$$

$$p(\mathbf{z}) = p(\mathbf{z}_L) \prod_{l=1}^{L-1} p(\mathbf{z}_l|\mathbf{z}_{l+1}), q(\mathbf{z}|\mathbf{x}) = q(\mathbf{z}_1|\mathbf{x}) \prod_{l=2}^{L} q(\mathbf{z}_l|\mathbf{z}_{l-1})$$

$$\mathbf{H} = (\mathbf{X}), \boldsymbol{\mu}, \boldsymbol{\sigma} = (\mathbf{H})$$

$$h_{out} = h_{in} + (((h_{in})))$$

$$\mathbf{z} = \boldsymbol{\mu} + \frac{\boldsymbol{\sigma}}{} \odot \boldsymbol{\epsilon}$$

$$\mathcal{L} = \|((\mathbf{z})) - \mathbf{z}\|_2^2$$

$$\mathbf{z}(t) = (1 - t) \cdot \mathbf{z}_1 + t \cdot \mathbf{z}_2, t \in [0, 1]$$

$$\beta(t) = \begin{cases} 0 & t < t \\ \beta \cdot \frac{t - t}{t - t} t \leq t \leq t \\ \beta & t > t \end{cases}$$

images/reparameterization_trick.png

images/residual_block_vae.png

images/temperature_sampling.png

images/mixed_precision_vae.png

images/latent_interpolation.png

images/onnx_pipeline.png

$\mathbf{QKV}$

$$(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \left(\frac{\mathbf{QK}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

d_k

$$(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = (_1, ..., _h)\mathbf{W}^O$$

$$_i = (\mathbf{QW}_i^Q, \mathbf{KW}_i^K, \mathbf{VW}_i^V)$$

$$\mathbf{A}_{i,j} = \mathbf{q}_i\mathbf{k}_j^T + \mathbf{q}_i\mathbf{r}_{i-j}^T + \mathbf{v}^T\mathbf{r}_{i-j}$$

$\mathbf{r}$

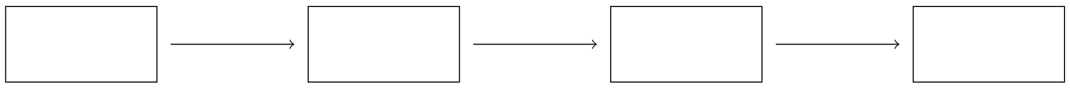
$$\mathbf{h}_{l+1} = \mathbf{h}_l + (((\mathbf{h}_l)))$$

$\mathcal{O}(T^2d)Td$

$\mathcal{O}(T^2)$

$$P(x_t) = \begin{cases} \frac{P(x_t)}{\sum_{x' \in V_k} P(x')} x_t \in V_k \\ 0 \end{cases}$$

$$V_p = \sum_{x \in V_p} P(x) \geq p$$



```
class PositionalEncoding(nn.Module):
    def __init__(self, embed_dim, max_len=2048):
        super().__init__()
        pe = torch.zeros(max_len, embed_dim)
        position = torch.arange(0, max_len, dtype=torch.float).unsqueeze
            (1)
        div_term = torch.exp(torch.arange(0, embed_dim, 2).float() *
                             (-math.log(10000.0) / embed_dim))

        pe[:, 0::2] = torch.sin(position * div_term)
        pe[:, 1::2] = torch.cos(position * div_term)

        self.register_buffer( pe , pe.unsqueeze(0))
        self.dropout = nn.Dropout(0.1)
```

```
self.section_memories = {}  # {section_id: memory_tensor}
```

```
def forward(self, x, use_memory=False):
    if use_memory and hasattr(self,  memory ) and self.memory is not
        None:
        x = torch.cat([self.memory, x], dim=1)

    # ... transformer processing ...

    if use_memory:
        max_memory_length = 1024
        self.memory = output.detach()
        if self.memory.size(1) > max_memory_length:
            self.memory = self.memory[:, -max_memory_length:, :]
```

generate_with_structure

←

```
# Top-k filtering
if top_k > 0:
    indices_to_remove = next_token_logits < torch.topk(next_token_logits
        , top_k)[0][..., -1, None]
    next_token_logits[indices_to_remove] = -float( inf )

# Nucleus (top-p) filtering
if top_p > 0.0:
    sorted_logits, sorted_indices = torch.sort(next_token_logits,
        descending=True)
    cumulative_probs = torch.cumsum(F.softmax(sorted_logits, dim=-1),
        dim=-1)
    sorted_indices_to_remove = cumulative_probs > top_p
    # Keep first token above threshold
    sorted_indices_to_remove[..., 1:] = sorted_indices_to_remove[...,
        :-1].clone()
    sorted_indices_to_remove[..., 0] = 0
```

```
encoder_layers = nn.TransformerEncoderLayer(
    d_model=embed_dim,
    nhead=num_heads,
    dim_feedforward=dim_feedforward,
    dropout=dropout,
    batch_first=True  # [batch, seq, features]
)
```

```
try:
    scripted_model = torch.jit.script(model.cpu())
    scripted_model.save("trained_transformer_jit.pt")
    print("JIT compiled model saved for faster inference")
except Exception as e:
    print(f"JIT compilation failed: {e}")
```

images/zha_transformer_arch.png

images/memory_mechanism.png

images/structured_generation.png

images/sampling_strategies.png

$$\mathbf{P}$$

$$p_{ij} = \frac{N_{ij}}{\sum_k N_{ik}} N_{ij} = (s_i \to s_j)$$

$$\lambda = 0,1$$

$$\hat{p}_{ij} = \frac{N_{ij} + \lambda}{\sum_k (N_{ik} + \lambda)}$$

$$= 2k - 2\ln(\hat{L})(k = , \hat{L} =)$$

$$\mathcal{L} = \underbrace{\|\mathbf{x} - (\mathbf{z})\|_2^2} + \beta \underbrace{D(\mathcal{N}(\mu^l)\sigma\mathcal{N}(0, \mathbf{I}))}$$

$$\beta$$

$$\theta, \phi \mathbf{X} \in \mathcal{D} \boldsymbol{\mu}, \boldsymbol{\sigma} \leftarrow {}_{\phi}(\mathbf{X}) \mathbf{z} \leftarrow \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}) \hat{\mathbf{X}} \leftarrow {}_{\theta}(\mathbf{z}) \theta, \phi \nabla_{\theta} \mathcal{L}, \nabla_{\phi} \mathcal{L}$$

$$\mathcal{L} = -\sum_{t=1}^{T} \log p(x_t|x_{<t})$$

$$\epsilon = 0,1$$

$\beta_1 = 0,9, \beta_2 = 0,98, \epsilon = 10^{-9}$

$$\eta_t = \eta + (\eta - \eta) \cdot (1, t/t)$$

$_memoriesastrukturltzeneiszakaszok(vers, refrn, hd)klnkezelsre.Ezlehetvtesziamodellszmra, hogym$

$_factor - rals10000 - szeresfinal_div_factor - ral.$

$_delta)meglltjaakpzst.Ezmegakadlyozzaatltanulstsoptimalizljaaszmtsierforrsokfelhasznlst.$

$_memoriesdictionaryklnkontextusttrolklnbzzeneiszakaszokhoz(vers, refrn, hd), lehetvtveakoherensis$

$_with_structuremetdusadaptvmemriakezelstalkalmaz, aholazismertszakaszokmemrijabetltsrekerl, mg$

$$_memory = TruebelltsgyorstjaazadattranszfertCPU - GPUkztt, mganon_blocking = Truelehetvtesziaztfedszmtstsadatmozgatst.$$

$\in \{16, 32, 64\} \beta \in [0.1, 1.0]$

$\in \{4, 6, 8\} \in \{4, 8, 12\}$

images/hyperparameter_optimization.png

images/pareto_optimization.png

10^{-3} 10^{-4}

$_interval(8-24)shiddenstates(8-32)szmokoptimalizlhatk[?].$

$_{f}actorbelltsoptimalizljaamemriahasznlatotsazadatbetltsisebessget.$

images/training_monitoring.png

images/debugging_tools.png

pipeline_diagram.png

$\rightarrow\rightarrow\rightarrow$

`data/raw/`

`data/processed/`

`markov_sample.npy`

`pretty_midi`

$$t = \text{round}\big(t \times 4\big) \,/\, 4.$$

`NOTE_ON_p``NOTE_OFF_p`$p \in [0, 127]$

`TIME_SHIFT_i`$i \in [1, 32]$

`VELOCITY_b`$b \in [1, 8]$

`CONTROL_SUSTAIN_ON``CONTROL_SUSTAIN_OFF` `src/models/markov.py`

$nn \leq 4$

`defaultdict(Counter)`

α

`START`

$P(\cdot \mid \text{context})$

`END`

$$h_t = \text{BiLSTM}(x_t, h_{t-1}), \mu = W_\mu \, h_T, \log \sigma^2 = W_\sigma \, h_T.$$

`src/models/vae.py`

$$t$$

$$\hat{x}_t = \mathrm{Softmax}\big(W_o[e_{t-1} \,\|\, z]\big),$$

$$e_{t-1} z$$

$\beta(e) = \min(1, e/E_{\text{warmup}})$

0.1

$\|\nabla\|_\infty \leq 1.0$

$d_{\text{model}} = 512$

$$B_{i,j}$$

src/trainers/TrainerBase

Hydra

matrix

```
for epoch in range(num_epochs):
    for batch in dataloader:
        recon, mu, logvar = model(batch)
        recon_loss = F.cross_entropy(recon, batch)
        kl = -0.5 * torch.sum(1 + logvar - mu.pow(2) - logvar.exp())
        loss = recon_loss + beta(epoch)*kl
        optimizer.zero_grad(); loss.backward()
        clip_grad_norm_(model.parameters(), 1.0)
        optimizer.step()
```

torch.cuda.amp

generate.py

```
def generate(length=512, top_p=0.9, temp=1.0):
    # Stage 1: Markov
    mk = MarkovModel.load( checkpoints/markov.npy )
    seq1 = mk.sample(length=128)

    # Stage 2: VAE
    vae = VAE.load( checkpoints/vae.pt )
    z = vae.encode(seq1)
    seq2 = vae.decode(z, temperature=temp)

    # Stage 3: Transformer
    trans = MusicTransformer.load( checkpoints/trans.pt )
    seq3 = trans.generate(prefix=seq2, max_len=length, top_p=top_p)

    midi = detokenize(seq3)
    midi.write( output/generated.mid )
```

$\approx$

$\approx$

$\approx$

$$p(x, z) = p(x \mid z)p(z)$$

x

z

$$\log p(x) = \log \int p(x, z)\, dz$$

$$q(z \mid x)$$

$$\begin{aligned} \log p(x) &= \log \int q(z \mid x) \frac{p(x, z)}{q(z \mid x)}\, dz \\ &\geq \int q(z \mid x) \log \frac{p(x, z)}{q(z \mid x)}\, dz = \mathcal{L}(q) \end{aligned}$$

$$\mathcal{L}(q) = \mathbb{E}_{q(z|x)}[\log p(x \mid z)] - (q(z \mid x) \,\|\, p(z))$$

$$\log p(x)$$

$q(z \mid x) p(z \mid x)$

$p(x \mid z)$

$$\theta \phi$$

$$z \sim q_\phi(z \mid x) z z = g_\phi(\epsilon, x) \epsilon \sim p(\epsilon)$$

$$\nabla_\phi \mathcal{L} \approx \nabla_\phi \mathbb{E}_{\epsilon \sim p(\epsilon)} \left[\log p_\theta(x \mid g_\phi(\epsilon, x)) - \log q_\phi(g_\phi(\epsilon, x) \mid x) \right]$$

$$\mathcal{L} \approx \frac{1}{L} \sum_{l=1}^{L} \left[\log p_\theta(x \mid z^{(l)}) - \log q_\phi(z^{(l)} \mid x) \right]$$

$$z^{(l)} = g_\phi(\epsilon^{(l)}, x) \epsilon^{(l)} \sim p(\epsilon)$$

$$h_t$$

$$h_t = \tanh(W_h h_{t-1} + W_x x_t + b)$$

$$\mathcal{L} W_h$$

$$\frac{\partial \mathcal{L}}{\partial W_h} = \sum_t \frac{\partial \mathcal{L}}{\partial h_t} \cdot \frac{\partial h_t}{\partial W_h}$$

$$\tfrac{\partial h_t}{\partial h_{t-1}}$$

$$\frac{\partial h_t}{\partial h_{t-k}} = \prod_{i=1}^{k} \frac{\partial h_{t-i+1}}{\partial h_{t-i}} = \prod_{i=1}^{k} W_h \cdot (1 - h_{t-i}^2)$$

$$W_h$$

W_h

tz_t

$$\mathcal{L} = \sum_{t=1}^{T} \mathbb{E}_{q(z_t \mid x_{\leq t})}[\log p(x_t \mid z_t, h_{t-1})] - (q(z_t \mid x_{\leq t}) \parallel p(z_t \mid h_{t-1}))$$

β

$$\mathcal{L}_\beta = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)] - \beta \cdot (q_\phi(z \mid x) \parallel p(z))$$

$\beta < 1$

$\beta = 1$

$\beta > 1$

z_i

$$\min_{\phi,\theta} \mathbb{E}_{p(x)}[-\mathcal{L}_\beta] = \min_{\phi,\theta} \mathbb{E}_{p(x)}[-\log p_\theta(x \mid z) + \beta \cdot (q_\phi(z \mid x) \parallel p(z))]$$

$$\mathcal{L} = \mathbb{E}_x \left[(|x_{i+1} - x_i|) \right]$$

$$x_i$$

$$\mathcal{L} = \mathcal{L}_\beta + \lambda \mathcal{L}$$

$$\begin{aligned} PE(pos, 2i) &= \sin\left(\frac{pos}{10000^{2i/d}}\right) \\ PE(pos, 2i+1) &= \cos\left(\frac{pos}{10000^{2i/d}}\right) \end{aligned}$$

$$|PE(pos, i)| \leq 1 PE(pos + k) PE(pos)$$

$$= (Q, [K; M_s], [V; M_s])$$

$$[K; M_s]$$

$$k$$

$$P(X_t = x_t \mid X_{t-1} = x_{t-1}, \ldots, X_1 = x_1) = P(X_t = x_t \mid X_{t-1} = x_{t-1}, \ldots, X_{t-k} = x_{t-k})$$

$$k$$

$$P_k(x_{t-k+1}^{t-1} \rightarrow x_t) = P(X_t = x_t \mid X_{t-1} = x_{t-1}, \ldots, X_{t-k+1} = x_{t-k+1})$$

$$|S| k |S|^k$$

$$\hat{P}_k(s \rightarrow s') = \frac{N(s \rightarrow s')}{\sum_{s''} N(s \rightarrow s'')}$$

$$N(s \rightarrow s') s s'$$

$$Z_t X_t t$$

$$P(X_{1:T}, Z_{1:T}) = P(Z_1) \prod_{t=2}^{T} P(Z_t \mid Z_{t-1}) \prod_{t=1}^{T} P(X_t \mid Z_t)$$

$P(Z_1)$

$P(Z_t \mid Z_{t-1})$

$P(X_t \mid Z_t)$

$$\mathbf{f} = [\mu, \sigma, R, \mu, \mu, \sigma, r]$$

μ, σ, R

μ

μ, σ

r

$$C_{ij} = \sum_{l=1}^{k} A_{il} B_{lj}$$

$$C_{ij}$$
$$O(n^2)O()$$

$$\mathcal{L} = S \cdot \mathcal{L}$$

$$\nabla = \frac{\nabla}{S}$$

$$BKb = B/K$$

$$\nabla\mathcal{L} = \frac{1}{B}\sum_{i=1}^{B}\nabla\mathcal{L}_i = \frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{b}\sum_{i\in\mathcal{B}_k}\nabla\mathcal{L}_i\right)$$

$$\mathcal{B}_k k$$

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min}) \left(1 + \cos\left(\frac{T}{T_{\max}}\pi\right)\right)$$

$$\eta_t = \begin{cases} \eta_{\min} + \frac{t}{t}(\eta_{\max} - \eta_{\min}) & t \leq t \\ \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min}) \left(1 + \cos\left(\frac{t-t}{T-t}\pi\right)\right) \end{cases}$$

$$_i = _{i+1} - _i$$

$$\pm$$

$$_e rssg)prokkntkdoldnak:$$

$$= \begin{cases} 2 \geq 0.9 \\ 10.4 \leq \ < 0.9 \\ 0 \end{cases}$$