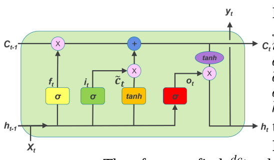
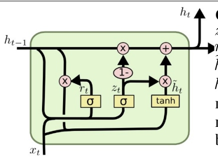
RNN: $h_t = \phi_h(W_h h_{t-1} + W_x x_t + b_h)$ and $y_t = \phi_t(W_y h_t + b_y)$ where ϕ is the activation.
Backward Pass: calculate $\mathcal{L}(\vec{\theta}) = L_{total}(\vec{\theta}) = \sum_{t=1}^{T} L(y_t)$ and $\frac{d}{d\theta}\mathcal{L}(\vec{\theta}) = \sum_{t=1}^{T} \frac{d}{d\theta} L(y_t)$ is computed for each $\theta \in \{W_h, W_x, W_y, b_h, b_y\}$.

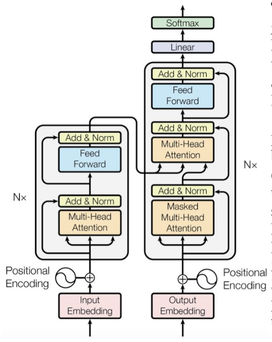
- $\frac{d\mathcal{L}(y_t)}{dW_y} = \frac{d\mathcal{L}(y_t)}{dy_t}\frac{dy_t}{dW_y}$ and $\frac{d\mathcal{L}(y_t)}{db_y} = \frac{d\mathcal{L}(y_t)}{dy_t}\frac{dy_t}{db_y}$
- $\frac{d\mathcal{L}(y_t)}{dW_x} = \frac{d\mathcal{L}(y_t)}{dy_t}\frac{dy_t}{dh_t}\frac{h_t}{dW_x}$ and $\frac{d\mathcal{L}(y_t)}{db_h} = \frac{d\mathcal{L}(y_t)}{dy_t}\frac{dy_t}{dh_t}\frac{dh_t}{db_h}$. However, W_x, b_h contribute to h_t directly and indirectly. Next bullet-point describes why we also need $\frac{dh_t}{dW_x} = \frac{\partial h_t}{\partial W_x} + \frac{dh_t}{dh_{t-1}}\frac{dh_{t-1}}{dW_x}$ and $\frac{dh_t}{db_h} = \frac{\partial h_t}{\partial b_h} + \frac{dh_t}{dh_{t-1}}\frac{dh_{t-1}}{db_h}$
- $\frac{d\mathcal{L}(y_t)}{dW_h} = \frac{d\mathcal{L}(y_t)}{dy_t}\frac{dy_t}{dh_t}\frac{h_t}{dW_h}$ The entries in the Jacobian $\frac{dh_t}{dW_h}$ contains the total gradient of $h_t[i]$ w.r.t $W_h[m,n]$. h_t depends on h_{t-1} which depends W_h. Therefore: $\frac{dh_t}{dW_h} = \frac{\partial h_t}{\partial W_h} + \frac{dh_t}{dh_{t-1}}\frac{dh_{t-1}}{dW_h}$
  If we continue expanding: $\frac{dh_t}{dW_h} = \frac{\partial h_t}{\partial W_h} + \frac{dh_t}{dh_{t-1}}(\frac{\partial h_{t-1}}{\partial W_h} + \frac{dh_{t-1}}{dh_{t-2}}\frac{dh_{t-2}}{dW_h}) = \sum_{\tau=1}^{t}(\prod_{l=\tau}^{t-1}\frac{dh_{l+1}}{dh_l})\frac{\partial h_\tau}{\partial W_h}$
  which results in the Back-propogation through time. We may truncate this to $\sum_{\tau=\max(1,t-L)}^{t}$.
  $\prod$ contains products of activation and weight matrix; gradient vanish/explode as $t \to \infty$!
  Therefore the long-term dependencies become harder to learn.



LSTM:
f_t : forget gate $= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
i_t : input gate $= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
o_t : output gate $= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$
$\tilde{c}_t$: candidate cell state $= \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$
c_t : memory cell state $= f_t \odot c_{t-1} + i_c \odot \tilde{c}_t$
h_t : hidden state $= o_t \odot \tanh(c_t)$
then, h_t : output of cell $= \phi_y(W_y h_y + b_y)$ simply.
Back propagation: we care about optimising the current state c_t. Therefore, we find $\frac{dc_t}{dW_c}$, ultimately this boils down to $\forall \tau \in \{0,\dots,t\}$ $\prod_{l=\tau}^{t-1}\frac{dc_{l+1}}{dc_l} = \prod_{l=\tau}^{t-1}[f_{l+1} + o_l \odot \frac{d\tanh(c_l)}{dc_l} \odot \frac{dc_{l+1}}{dh_l}]$. This deals with the exploding gradient issue, as the gradient contains terms proportional to $f_{i+1}\prod^{i-1} l = 1 o_l \odot \frac{dc_{l+1}}{dh_l} \approx 0$ when $f_{i+1} \approx 0$ with $\forall i \in \{\tau+1,\dots,t-1\}$. Similarly, this alleviates gradient since the term is proportional to $\prod_{l=\tau+1}^{i} f_l \odot o_\tau \odot \frac{dc_{\tau+1}}{dh_\tau} \approx o_\tau \odot \frac{dc_{\tau+1}}{dh_\tau}$ when $f_l \to 1$ with $\forall l \in \{\tau+1,\dots,i\}$; TLDR; the gates can switch on or off the relevant terms in gradient computation. **Struggles learn long-term deps.**



Gated Recurrent Unit:
z_t : switch gate $= \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$
r_t : reset gate $= \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$
$\tilde{h}_t$: candidate hidden state $= \tanh(W_h \cdot [r_t \cdot h_{t+1}, x_t] + b_h)$
h_t : hidden state $= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$
no longer has input/output gate but maintains forgetting mechanism. GRU more efficient computing & less overfitting but LSTM good default choice.



Transformer Architecture: This model outputs $p_\theta(y_l|y_{<l}, z_{1:T})$ with $y_{<l}$being masked out in the first layer of the decoder, and $z_{1:T}$ encoder attention outputs used in decoder. The input embedding receives $x_{1:T}$ and the output embedding receives $y_{1:L}$
Encoder:
1) we obtain word embeddings and concatenate with positional encodings (which are also learnt). This is because single head attention is permutation equivariant (if we swap two rows in the query matrix then the attention output also has these two rows swapped). Thus, we break permutation equivariance by learning or pre-defined mapping. If we know max length then learnt positional encodings may be better.
2) norm is used within a single hidden layer output
3) the output of multi-head attention is further processed by a feed forward network which is applied to each row of the matrix.

Decoder:
1) after embedding and positional encoding, masked attention is used so during training only the previous words can be used.
2) The input representation is fed into the multi-head attention. The encoder output $z_{1:T}$ is used as the keys and values of the attention module. This allows the decoder to attend every word in the input $x_{1:T}$ for each of the predicted outputs $y_{1:l-1}$ so far.
The network is trained using maximum likelihood or cross entropy loss.

Sequence to Sequence (Seq2Seq) Auto-regressive Model: given two varying sentence length pairs, $p_\theta(y_{1:L}|x_{1:T}) = \prod_{l=1}^{L} p_\theta(y_l|y_{<l}, v), \quad v = enc(x_{1:T})$ fit an LSTM model 'sequence decoder' with $p_\theta(y_l|y_{<l}, v) = p_\theta(y_l|h_l^d, c_l^d), \quad h_l^d, c_l^d = LSTM_\theta^{dec}(y_{<l})$ where the first hidden states h_0^d, c_0^d are initialised by an encoder $v = enc(x_{1:T}) = NN_\theta(h_T^e, c_T^e), \quad h_T^e, c_T^e = LSTM_\theta^{enc}(x_{1:T})$. We initialise the system by encoding the entirety of the x input. This LSTM will return the first word (upon seeing the <eos> and use the hidden state v to initialise the decoder LSTM. Alternatively, the first word can be an <sos> token. This can be extended to image captioning where we extract an image encoding with CNNs and feed it into an LSTM.
Generative Models for Sequence VAE: We use an encoder architecture like an LSTM to predict the mean and standard variation of the q distribution. The loss function is defined similarly to before:

$$\boldsymbol{\phi}^*, \boldsymbol{\theta}^* = \arg\max \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}), \quad \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}) = \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}_{1:T})}[\underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_{1:T})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:T}|\boldsymbol{z})] - \text{KL}[q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_{1:T})||p(\boldsymbol{z})]}_{:=\mathcal{L}(\boldsymbol{x}_{1:T}, \boldsymbol{\phi}, \boldsymbol{\theta})}].$$

Encoder: $q_\phi(z|x_{1:T}) = N(z; \mu_\phi(x_{1:T}), diag(\sigma_\phi^2(x_{1:T})))$, where $\mu_\phi(x_{1:T}), \log\sigma_\phi(x_{1:T}) = LSTM_\phi(x_{1:T})$ and the Generator: $p_\theta(x_{1:T}|z) = \prod_{t=1}^{T} p_\theta(x_t|x_{<t}, z)$.

State-Space-Model: stochastic dynamic model for the latent space $p_\theta(z_{1:T}) = p_\theta(z_1)\prod_{t=2}^{T} p_\theta(z_t|z_{t-1})$. The emission depends on zt only: $p_\theta(x_t|z_t)$. The joint distribution is thus $p_\theta(x_{1:T}, z_{1:T}) = \prod_{t=1}^{T} p_\theta(z_t|z_{t-1})p_\theta(x_t|z_t)$
Hidden Markov Model (HMM) $z_t = Az_{t-1} + B\epsilon_t, \epsilon_t \sim N(0, I)$ with a Linear Gaussian emission model $x_t = Cz_t + D\psi_t, \psi_t \sim \epsilon_t$. **State-space models + non-linear dynamics**: embed an LSTM which will predict the mean and variance for the next latent variable z at t+1: $p_\theta(y_t|z_t) : x_t = \mu_\theta^x(z_t) + \sigma_\theta^x(z_t)\psi_t, \psi_t \sim N(0, I)$ with the joint distribution $p_\theta(x_{1:T}, z_{1:T}) = \prod_{t=1}^{T} p_\theta(z_t|z_{t-1})p_\theta(x_t|z_t)$. We train it with

$$E_{p_{data}(x_{1:T})}[\log p_\theta(x_{1:T})] \geq E_{p_{data}(x_{1:T})}[E_{q_\phi(z_{1:T}|x_{1:T})}[\log p_\theta(x_{1:T}|z_{1:T})] - KL[q_\phi(z_{1:T}|x_{1:T})||p_\theta(z_{1:T})]]$$

For the encoder, we can base the q distribution based on the current input x with: $q_\phi(z_{1:T}|x_{1:T}) = \prod_{t=1}^{T} q_\phi(z_t|x_{\leq t})$ where $q_\phi(z_t|x_{\leq t}) = N(z_t; \mu_\phi^z(h_t^e), diag(\sigma_\phi^z(h_t^e)^2)), [h_t^e, c_t^e] = LSTM_\phi(x_t, h_{t-1}^e, c_{t-1}^e)$

Attention: before, we use v (the hidden state from the last LSTM block). How, use v_l such that with attention: $p_\theta(y_{1:L}|x_{1:T}) = \prod_{l=1}^{L} p_\theta(y_l|y_{<l}, v_l)$ with $v_l = \sum_{t=1}^{T}\alpha_{lt}f_t$ which are the features each weighted by attention α_{lt} and $h_l^d, c_l^d = LSTM_\theta^{dec}([y_{l-1}, v_l], h_{l-1}^d, c_{l-1}^d)$ where $f_t = T_\theta(h_t^e)$ (feature output of the encoder at time t) and similarity $e_{lt} = a(h_{l-1}^d, f_t)$ and $a_l = softmax(e_l), e_l = \{e_{l1}, \dots, e_{lT}\}$

$$Attention(Q, K, V; a) = a\left(\frac{QK^\top}{\sqrt{d_q}}\right)V$$
$$Q = (\boldsymbol{q}_1, \dots, \boldsymbol{q}_N)^\top \in \mathbb{R}^{N \times d_q}$$
$$K = (\boldsymbol{k}_1, \dots, \boldsymbol{k}_M)^\top \in \mathbb{R}^{M \times d_q}$$
$$V = (\boldsymbol{v}_1, \dots, \boldsymbol{v}_M)^\top \in \mathbb{R}^{M \times d_v}$$
a(.) activation function applied row-wise

Single Head Attention: each key is paired with a value in the V matrix. The query matrix contains query matricies. Hard attention is when $a = onehot(\arg\max x_i)$ and Soft attention is when $a = softmax(x)$ Masked attention is when we mask out some attention values (mask is 0 or 1 and sets the cell in the $QK^\top$ matrix as $-\infty$ (useful for sequence prediction with a given ordering: in test time "future" is not available for the "current" to attend).
Complexity: *time* $O(MNd_q + MNd_v)$, *space* $O(MN + Nd_v)$, # params (only key and value) so $K, V : O(Md_q + Md_v)$ and in self attention this is $O(Nd_v)$ i.e. Q=V=K

Multi Head Attention: $Multihead(Q, K, V, a) = concat(head_1, \dots, head_n)W^O$, where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V, a)$ and Attention is defined as before. This helps define different hidden similarity measures. **Complexity**: $W_i^Q \in \mathbb{R}^{d_q \times d_q}, W_i^K \in \mathbb{R}^{d_q \times \tilde{d}_q}, W_i^V \in \mathbb{R}^{d_v \times \tilde{d}_v}, W^O \in \mathbb{R}^{h\tilde{d}_v \times d_{out}}$ then
time complexity $O(\underbrace{h(MN\tilde{d}_q + MN\tilde{d}_v)}_{\text{attention heads}} + \underbrace{h(\tilde{d}_q d_q(M+N) + \tilde{d}_v d_v M)}_{\text{input projections}} + \underbrace{Nh\tilde{d}_v d_{out}}_{\text{combined outputs}})$
space complexity $O(\underbrace{h(MN + N\tilde{d}_v)}_{\text{attention heads}} + \underbrace{h(\tilde{d}_q(M+N) + \tilde{d}_v M)}_{\text{input projections}} + \underbrace{Nd_{out}}_{\text{combined outputs}})$ where to keep complexity similar we set $\tilde{d}_q = \left\lfloor \frac{d_q}{h} \right\rfloor, \tilde{d}_v = \left\lfloor \frac{d_v}{h} \right\rfloor$ to keep the costs close to performing single-head attention in the original space.