CS 480/680

3/12/26

# Lecture 17

--- Auto Encoders
Gen Models

.... Transformers

Q3 = week 11

$X, Z \rightarrow p_\theta(X,Z) = p_\theta(Z)\, p_\theta(X|Z) =$ $p_\theta(X)\, p_\theta(Z|X)$

$p_\theta(Z) \sim N(0,I)$ — Decoder

$p_\theta(X) = \int_{\mathbb{R}^m} p_\theta(X,Z)\, dz$

$$p_\theta(x) = \frac{p_\theta(x,z)}{p_\theta(z|x)} \quad \leftarrow \approx p_\varphi(Z|X) \quad \text{Encoder} \qquad 3.1$$

Idea 3 "Variational approximation"

3.2 — data point $x \in \mathcal{D}$

$$\ln p_\theta(x) = E_\varphi[\ln p_\theta(x)] = E_\varphi\left[\ln \frac{p_\theta(x,z)}{p_\theta(z|x)} \cdot \frac{p_\varphi(z|x)}{p_\varphi(z|x)}\right] =$$

$E_\varphi \equiv E_{z \sim p_\varphi(z|x)}$

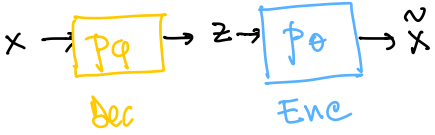
log-l

$$= E_\varphi[\ln p_\theta(x,z) - \ln p_\varphi(z|x)] + E_\varphi[\ln p_\varphi(z|x) - \ln p_\theta(z|x)]$$

$p(z)$

Evidence Lower BOund (ELBO)

$KL(p_\varphi(z|x) \,\|\, p_\theta(z|x)) \geq 0$ ← Ex




Why sampling?

Likelihood: • $\int$ → Monte Carlo

Integral ≈ Avg.

$$\int_a^b f(u)\, p(u)\, du \approx \frac{1}{N} \sum_{i=1}^{N} f(u^i)$$

$u^i \sim p(U) \quad i = 1{:}N$

Gradient • $\frac{\partial \ln p_\theta}{\partial \theta, \varphi} \approx$ avg $\Leftrightarrow$ SGD

$$\text{ELBO} = E_\varphi\left[\ln p_\theta(x,z) - \ln p_\varphi(z|x)\right] = E_\varphi\left[\ln p_\theta(x|z)\right] - E_\varphi\left[\ln p_\varphi(z|x) - \ln p(z)\right]$$

KL div

3.2: $p_\theta(x,z) = p_\theta(x|z)\cdot p(z)$, $N(0,I)$

3.3 sample $\varepsilon \sim N(0, I_m) \rightarrow z_{(\varepsilon)} = \varepsilon \cdot \sigma_{(x)} + \mu_{(x)}$

↑ sample $n_\varepsilon$ values

write this rigorously

$z|x \sim N(\mu_{(x)}, \sigma^2_{(x)} I) = \prod_{j=1}^{m} N(\mu_j, \sigma_j^2)$

$$p(z|x) = \frac{e^{\frac{-\|z-\mu\|^2}{2\sigma^2}}}{(2\pi)^{m/2}\prod\sigma_j}$$

$$\sigma = \begin{bmatrix}\sigma_1\\ \vdots \\ \sigma_m\end{bmatrix} \quad \mu = \begin{bmatrix}\mu_1\\ \vdots \\ \mu_m\end{bmatrix}$$

$$\text{ELBO} \approx \frac{1}{n_\varepsilon}\sum_\varepsilon \left\{ \ln p_\theta(x|z_{(\varepsilon)}) + \sum_j \left(\ln \sigma_j^2 \cdot \frac{1}{2} + \ln \sigma_j\right)\right\}$$

$\ln p(z|x)$

$$-\frac{1}{2}\sum_j (\varepsilon_j \sigma_j + \mu_j)^2 \Big\} \qquad \varepsilon \sim$$

$$\frac{z_j - \mu_j}{\sigma_j} = \varepsilon_j$$

$$\ln p(z|x) = -\frac{1}{2}\sum_{j=1}^{m}\left(\varepsilon_j^2 + \ln \sigma_j\right) + \text{ct}$$

$$\ln p(z) = -\frac{\|z\|^2}{2} + \text{ct}$$

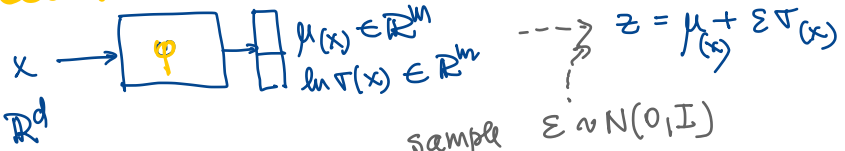
$$= -\frac{1}{2}\sum_j (\mu_j + \sigma_j \varepsilon_j)^2$$

$dz_1 \cdots dz_m$

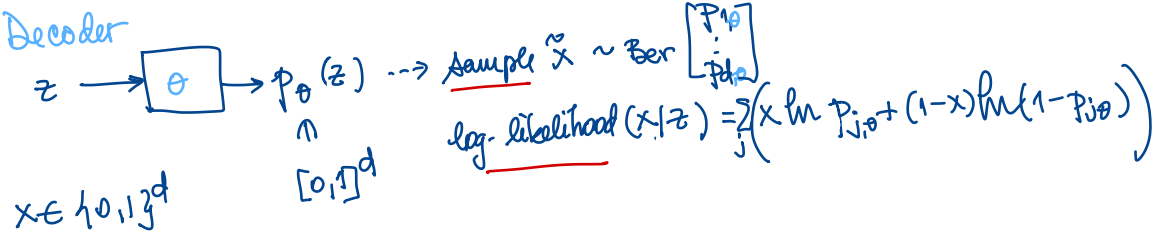
$$p_\varphi(z)dz = p_\varphi(z(\varepsilon))\cdot \frac{dz}{d\varepsilon}\cdot d\varepsilon$$

$$\int_{\mathbb{R}^m} f(z)\, p_\varphi(z)\, dz = \int_{\mathbb{R}^m} f(z(\varepsilon))\, p_\varphi(z(\varepsilon)) \prod_{j=1}^{m}\sigma_j \, d\varepsilon_1 \cdots d\varepsilon_m \, d\varepsilon$$

# Encoder




# Decoder



# Summary

- Solve generative modelling
- Use neural network to map Gaussian samples to data distribution
- Do it by using variational autoencoder: tries to map original distribution to a Gaussian, and also maps back to original distribution. Each is encoded in the loss function.

# Samples from a VAE

(a) 2-D latent space

(b) 5-D latent space

(c) 10-D latent space

$m = 10$ $z$

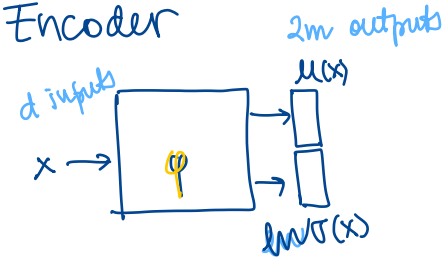
(d) 20-D latent space

Note $z_j = \mu_j + \sigma_j \varepsilon_j$

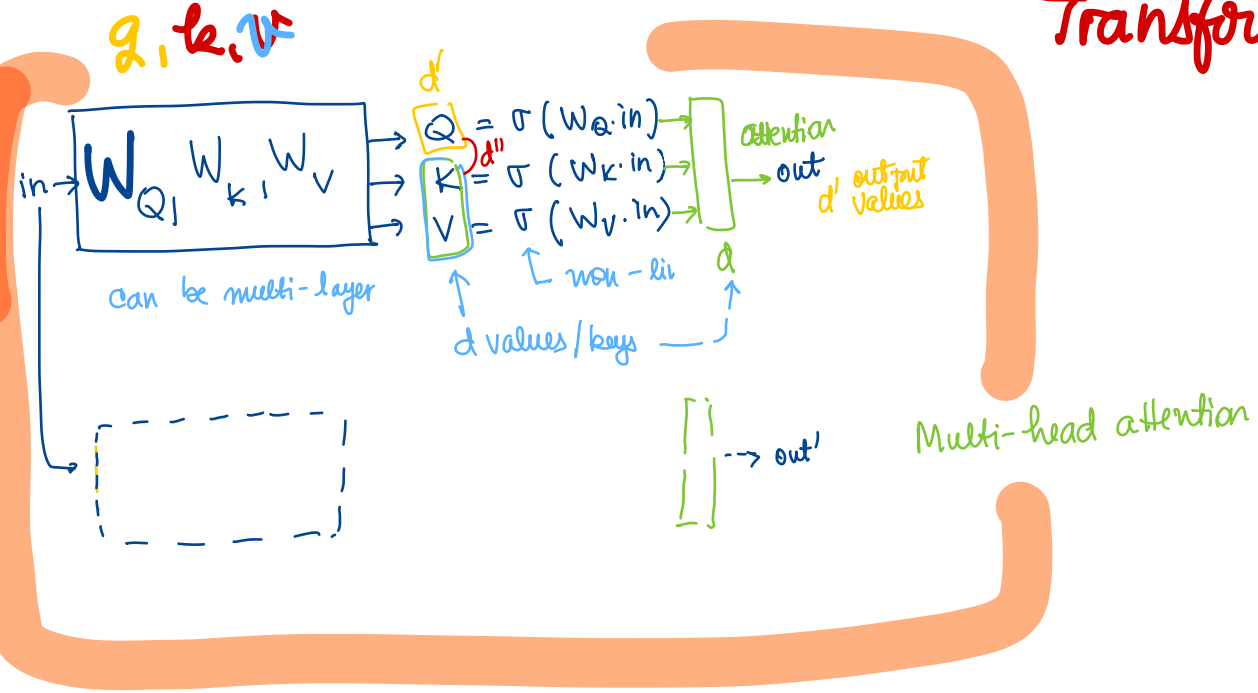
(c) $\dfrac{z_j - \mu_j}{\sigma_j} = \varepsilon_j \Rightarrow$ indep of $\sigma_\varphi, \mu_\varphi$!

(e) $\|z\|^2 = \sum_{j=1}^{m} (\mu_j + \sigma_j \varepsilon_j)^2 \Rightarrow$ depends on $\varphi$ !

Encoder

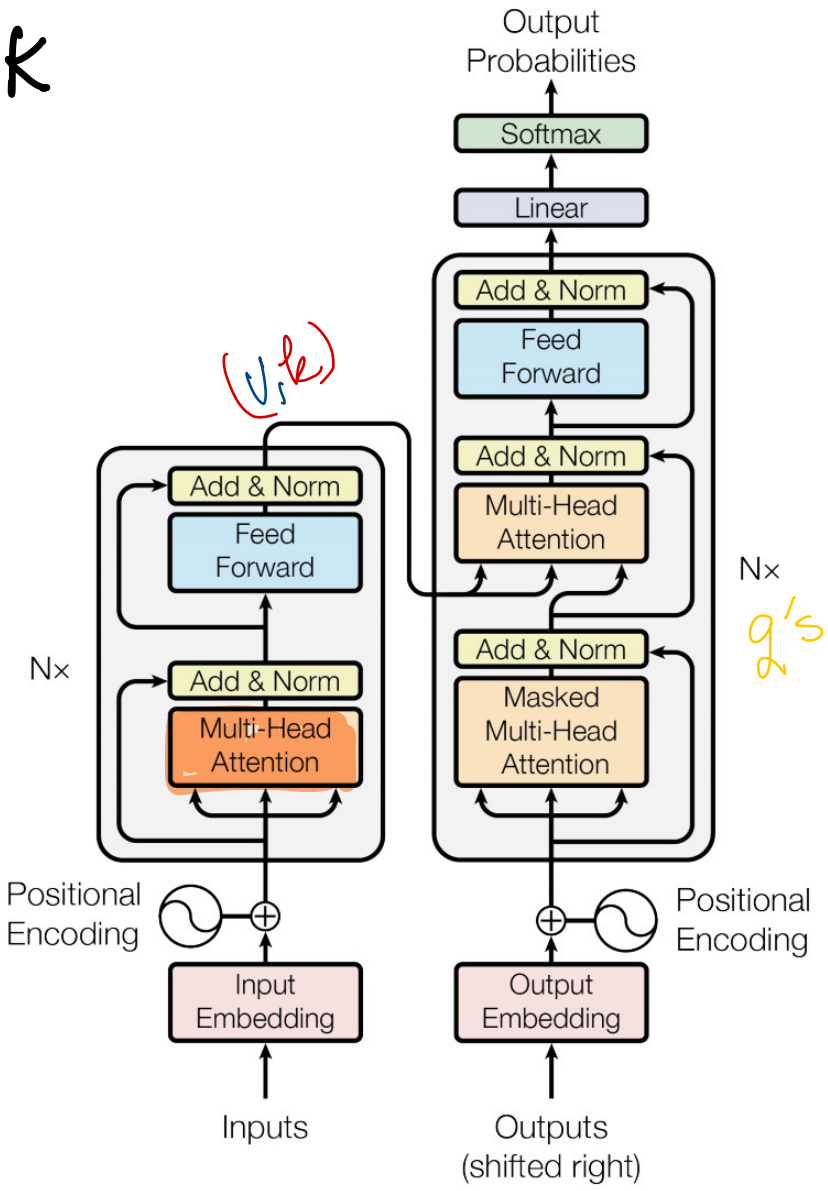


# Transformers



# Transformer Network

- Vaswani et al., (2017) Attention is all you need.
- Encoder-decoder based on attention (no recurrence)



# Masked Multi-head attention

- Masked multi-head attention: multi-head where some values are masked (i.e., probabilities of masked values are nullified to prevent them from being selected).
- When decoding, an output value should only depend on previous outputs (not future outputs). Hence we mask future outputs.

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

$$maskedAttention(Q, K, V) = softmax\left(\frac{Q^T K + M}{\sqrt{d_k}}\right)V$$

where $M$ is a mask matrix of 0's and $-\infty$'s

# Other layers

- Layer normalization:
  - Normalize values in each layer to have 0 mean and 1 variance
  - For each hidden unit $h_i$ compute $h_i \leftarrow \frac{g}{\sigma}(h_i - \mu)$

    where $g$ is a variable, $\mu = \frac{1}{H}\sum_{i=1}^{H} h_i$ and $\sigma = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(h_i - \mu)^2}$
  - This reduces "covariate shift" (i.e., gradient dependencies between each layer) and therefore fewer training iterations are needed

- Positional embedding (embedding to distinguish each position):

$$PE_{position,2i} = \sin(position/10000^{2i/d})$$
$$PE_{position,2i+1} = \cos(position/10000^{2i/d})$$

# Comparison

- Attention reduces sequential operations and maximum path length, which facilitates long range dependencies

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. $n$ is the sequence length, $d$ is the representation dimension, $k$ is the kernel size of convolutions and $r$ the size of the neighborhood in restricted self-attention.

| Layer Type | Complexity per Layer | Sequential Operations | Maximum Path Length |
|---|---|---|---|
| Self-Attention | $O(n^2 \cdot d)$ | $O(1)$ | $O(1)$ |
| Recurrent | $O(n \cdot d^2)$ | $O(n)$ | $O(n)$ |
| Convolutional | $O(k \cdot n \cdot d^2)$ | $O(1)$ | $O(log_k(n))$ |
| Self-Attention (restricted) | $O(r \cdot n \cdot d)$ | $O(1)$ | $O(n/r)$ |

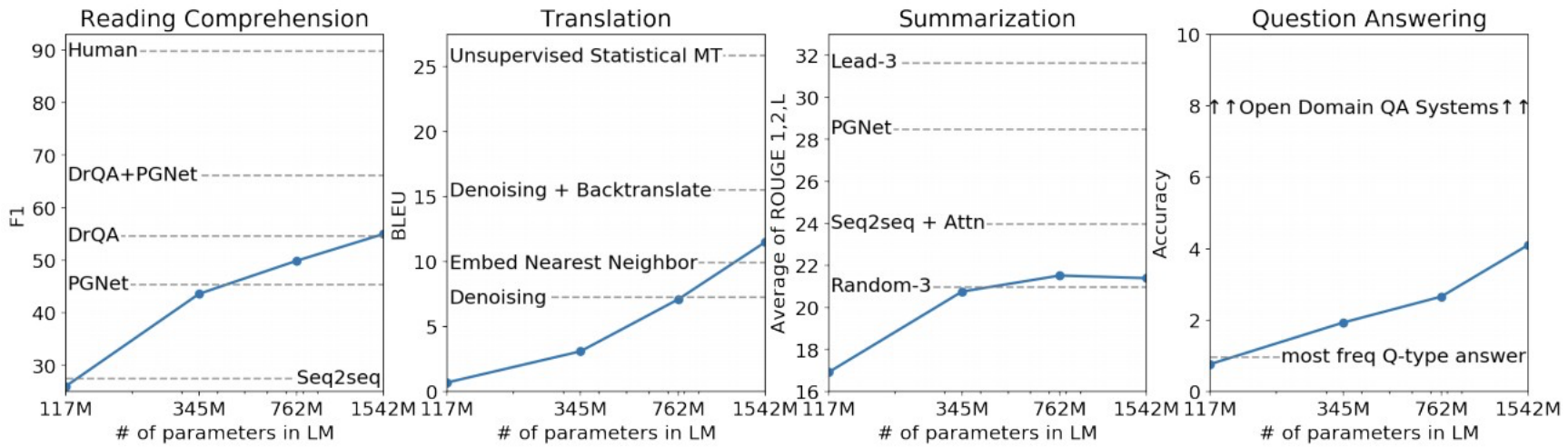
# Results

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

| Model | BLEU | | Training Cost (FLOPs) | |
|---|---|---|---|---|
| | EN-DE | EN-FR | EN-DE | EN-FR |
| ByteNet [15] | 23.75 | | | |
| Deep-Att + PosUnk [32] | | 39.2 | | $1.0 \cdot 10^{20}$ |
| GNMT + RL [31] | 24.6 | 39.92 | $2.3 \cdot 10^{19}$ | $1.4 \cdot 10^{20}$ |
| ConvS2S [8] | 25.16 | 40.46 | $9.6 \cdot 10^{18}$ | $1.5 \cdot 10^{20}$ |
| MoE [26] | 26.03 | 40.56 | $2.0 \cdot 10^{19}$ | $1.2 \cdot 10^{20}$ |
| Deep-Att + PosUnk Ensemble [32] | | 40.4 | | $8.0 \cdot 10^{20}$ |
| GNMT + RL Ensemble [31] | 26.30 | 41.16 | $1.8 \cdot 10^{20}$ | $1.1 \cdot 10^{21}$ |
| ConvS2S Ensemble [8] | 26.36 | **41.29** | $7.7 \cdot 10^{19}$ | $1.2 \cdot 10^{21}$ |
| Transformer (base model) | 27.3 | 38.1 | **$3.3 \cdot 10^{18}$** | |
| Transformer (big) | **28.4** | **41.0** | $2.3 \cdot 10^{19}$ | |

# GPT and GPT-2

- Radford et al., (2018) Language models are unsupervised multitask learners
  - Decoder transformer that predicts next word based on previous words by computing $P(x_t|x_{1..t-1})$
  - SOTA in "zero-shot" setting for 7/8 language tasks (where zero-shot means no task training, only unsupervised language modeling)



# BERT (Bidirectional Encoder Representations from Transformers)

- Devlin et al., (2019) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
  - Decoder transformer that predicts a missing word based on surrounding words by computing $P(x_t|x_{1..t-1,t+1..T})$
  - Mask missing word with masked multi-head attention
  - Improved state of the art on 11 tasks

| System | MNLI-(m/mm) 392k | QQP 363k | QNLI 108k | SST-2 67k | CoLA 8.5k | STS-B 5.7k | MRPC 3.5k | RTE 2.5k | **Average** - |
|---|---|---|---|---|---|---|---|---|---|
| Pre-OpenAI SOTA | 80.6/80.1 | 66.1 | 82.3 | 93.2 | 35.0 | 81.0 | 86.0 | 61.7 | 74.0 |
| BiLSTM+ELMo+Attn | 76.4/76.1 | 64.8 | 79.8 | 90.4 | 36.0 | 73.3 | 84.9 | 56.8 | 71.0 |
| OpenAI GPT | 82.1/81.4 | 70.3 | 87.4 | 91.3 | 45.4 | 80.0 | 82.3 | 56.0 | 75.1 |
| $\text{BERT}_{\text{BASE}}$ | 84.6/83.4 | 71.2 | 90.5 | 93.5 | 52.1 | 85.8 | 88.9 | 66.4 | 79.6 |
| $\text{BERT}_{\text{LARGE}}$ | **86.7/85.9** | **72.1** | **92.7** | **94.9** | **60.5** | **86.5** | **89.3** | **70.1** | **82.1** |

# Limitation

- Transformers scale quadratically with sequence length
  - In practice, sequence length often limited to 512 tokens

- How can we process long sequences?
  - Hierarchy of transformers (i.e., words→sentences→documents→corpus)
  - Approximate transformers (i.e., longformer, reformer, performer, etc.)
  - **Structured State Space Sequence (S4) model**

- S4: Very recent approach (Gu, Goel & Re, ICLR 2022)
  - Potential to displace transformers
  - S4 achieved state of the art on Long Range Arena benchmark
  - Scales linearly with sequence length