CSE 480/680

3/12/26

# Lecture 17

- Transformers
- VAE + Gen Models

F : April 18

LVI → VAE t.bp.

Q3 = Week 11

A7 : Week 10

[A6] < A7

NOT graded

# Attention Mechanism

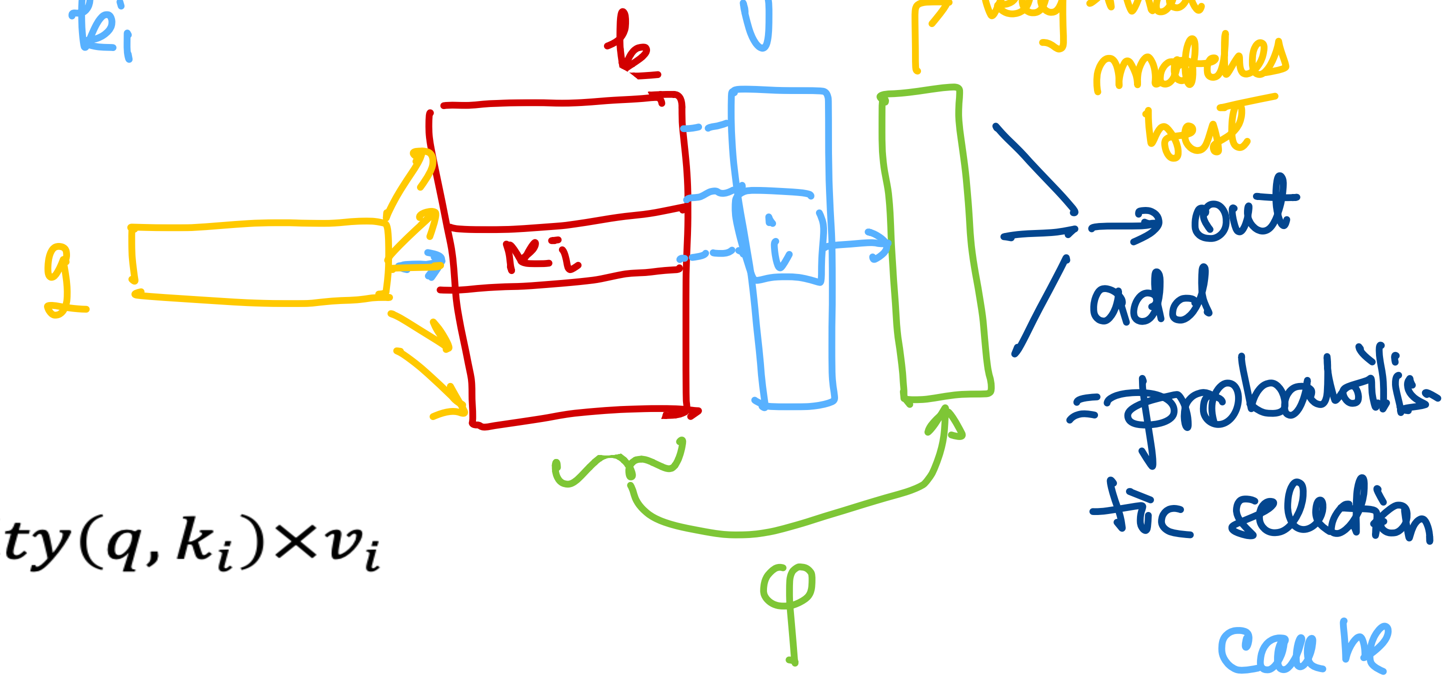
$\varphi = \text{softmax}(q^T k_i,\ i=1:d)$

- Mimics the retrieval of a $\boxed{\textbf{value } v_i}$ for a **query** $q$ = in x based on a **key** $k_i$ in database

out

$\downarrow$ $k_i$

- Picture



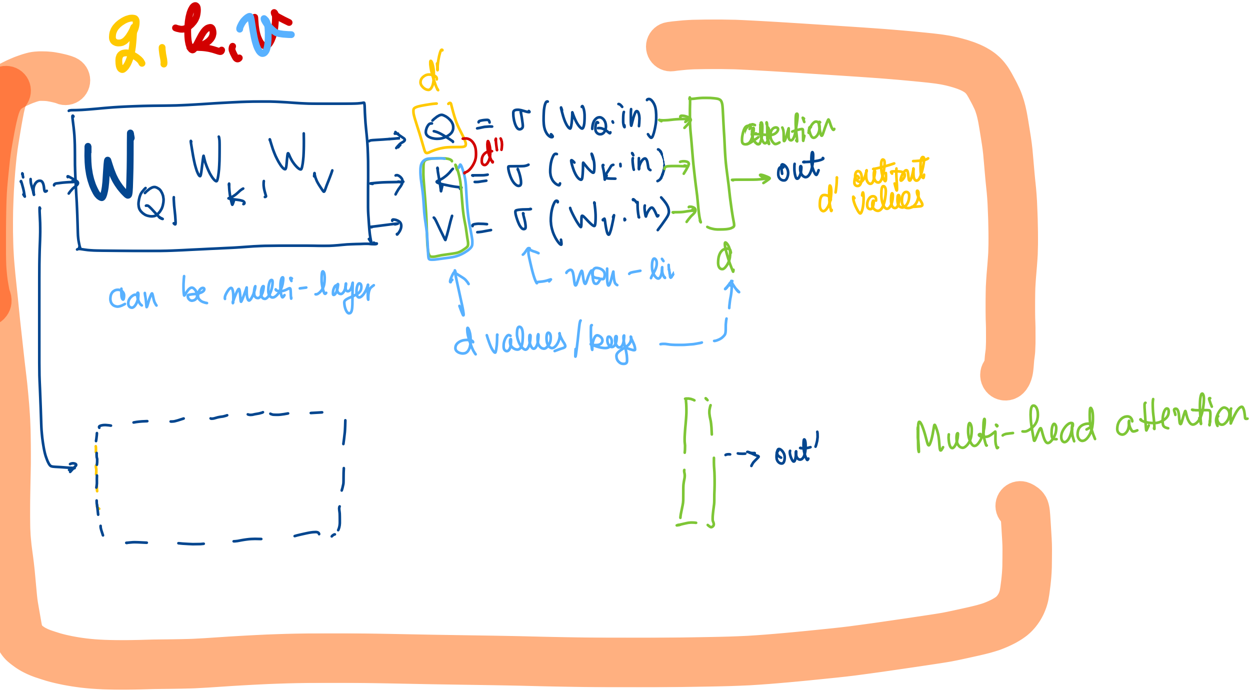
retrieval
||

$$attention(q, \boldsymbol{k}, \boldsymbol{v}) = \textstyle\sum_i similarity(q, k_i) \times v_i$$

extract value

$\varphi = [0.1\quad 0.001\quad .899\quad 0\quad 0] \Rightarrow out = \underline{v_3} \times .899 + v_1 \times .1 + v_2 \cdot .001$

can be a vector

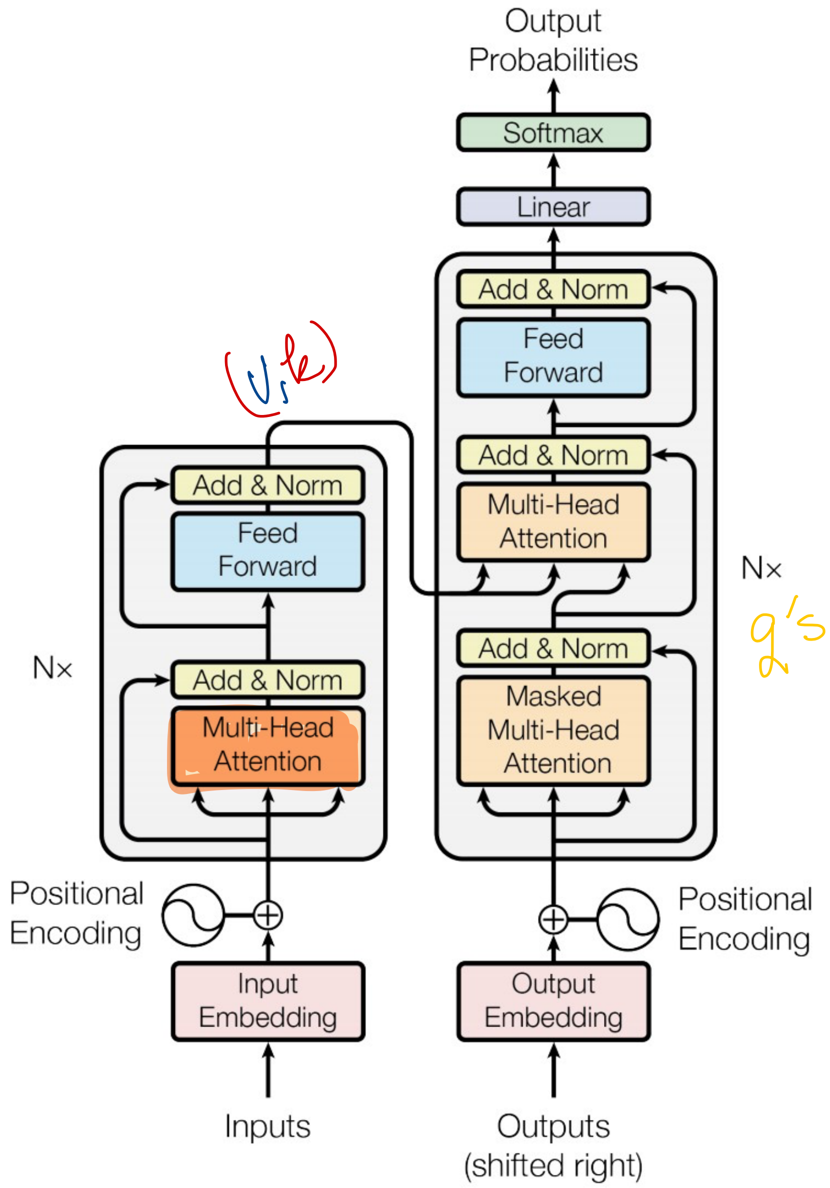

q, k, v
in
$W_Q, W_K, W_V$
can be multi-layer
d'
d''
$Q = \sigma(W_Q \cdot in)$
$K = \sigma(W_K \cdot in)$
$V = \sigma(W_V \cdot in)$
non-lin
d values / keys
attention
out
d' output values
d
out'
Multi-head attention

# Attention Mechanism (Neural Architecture)

- Example: machine translation
  - Query: $s_{i-1}$ (hidden vector for $i-1^{th}$ output word)
  - Key: $h_j$ (hidden vector for $j^{th}$ input word)
  - Value: $h_j$ (hidden vector for $j^{th}$ input word)

# Transformer Networ

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



# Multihead attention

- Multihead attention: compute multiple attentions per query with different weights

$$multihead(Q, K, V) = W^O concat(head_1, head_2, \dots, head_h)$$
$$head_i = attention\left(W_i^Q Q, W_i^K K, W_i^V V\right)$$
$$attention(Q, K, V) = softmax\left(\frac{Q^T K}{\sqrt{d_k}}\right) V$$

# 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

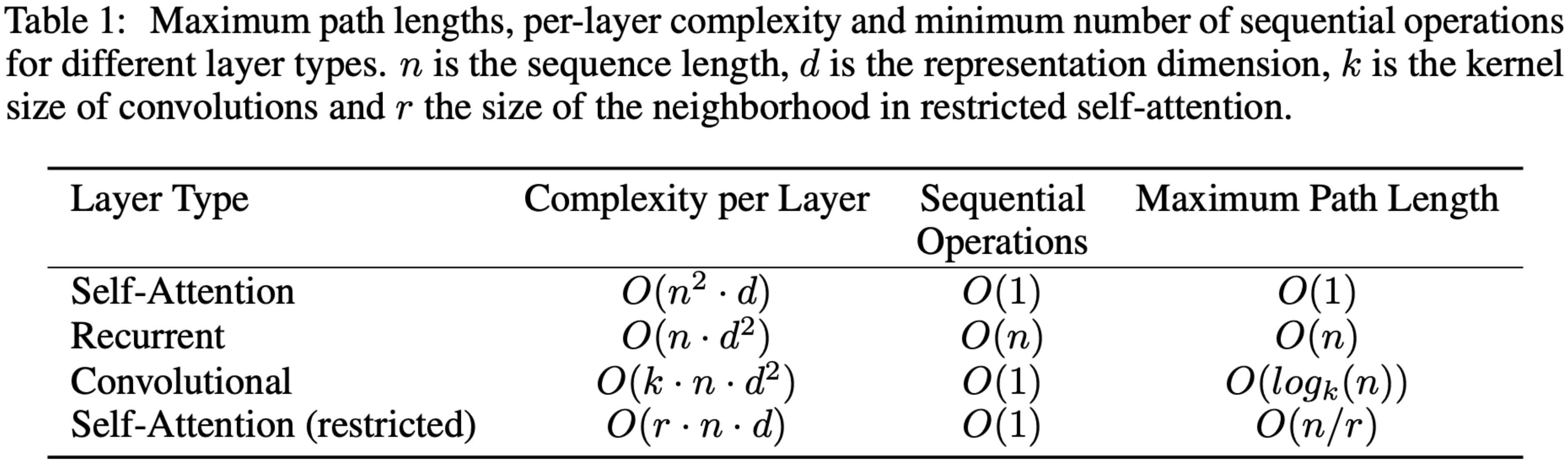


Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. $n$ is the sequence length, $d$ is the representation dimension, $k$ is the kernel size of convolutions and $r$ the size of the neighborhood in restricted self-attention.

| 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

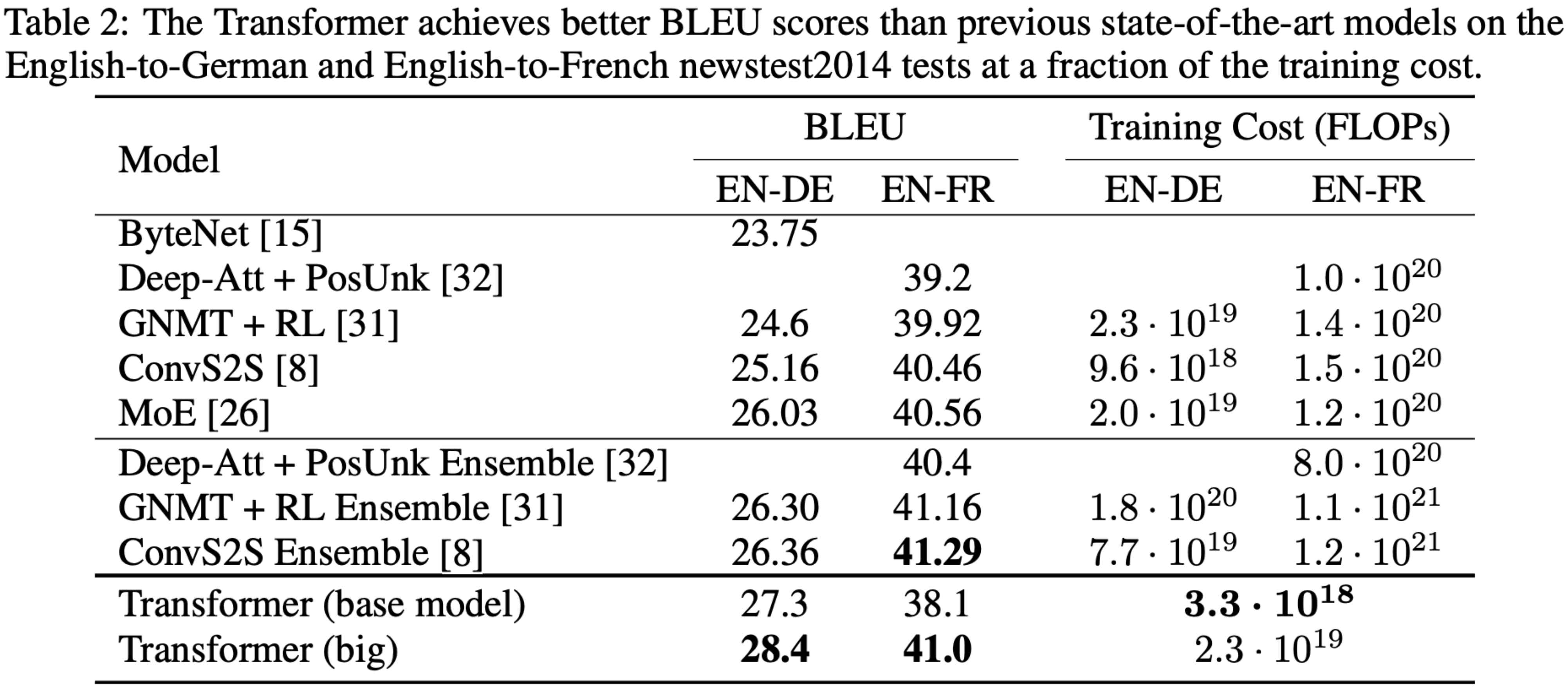
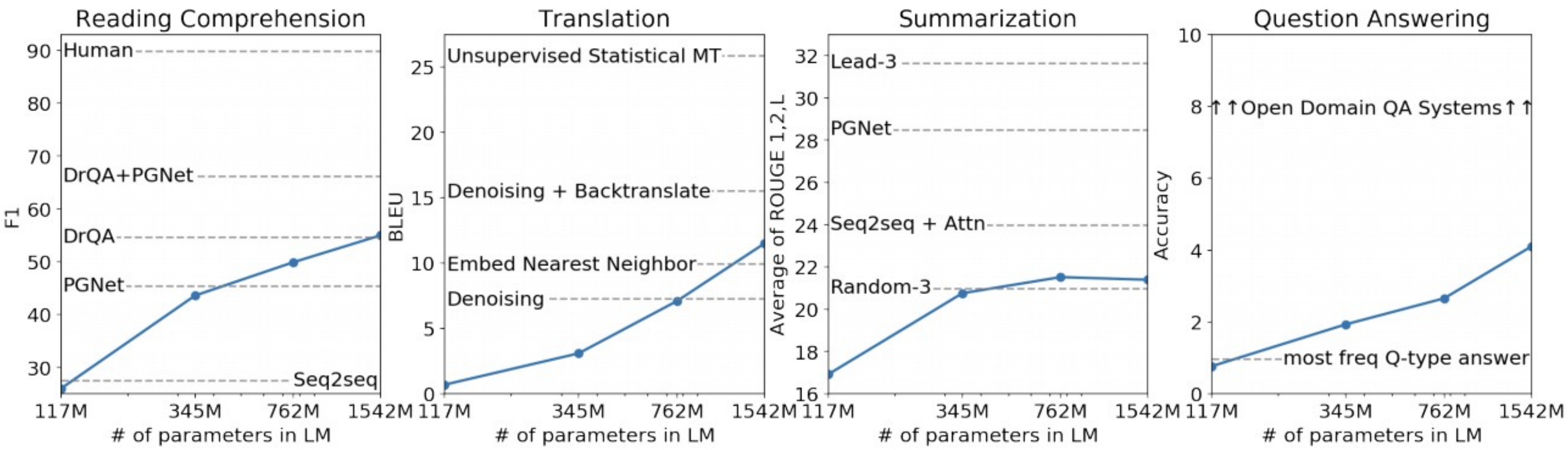


Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

| 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 | $\mathbf{3.3 \cdot 10^{18}}$ | |
| Transformer (big) | **28.4** | **41.0** | $2.3 \cdot 10^{19}$ | |

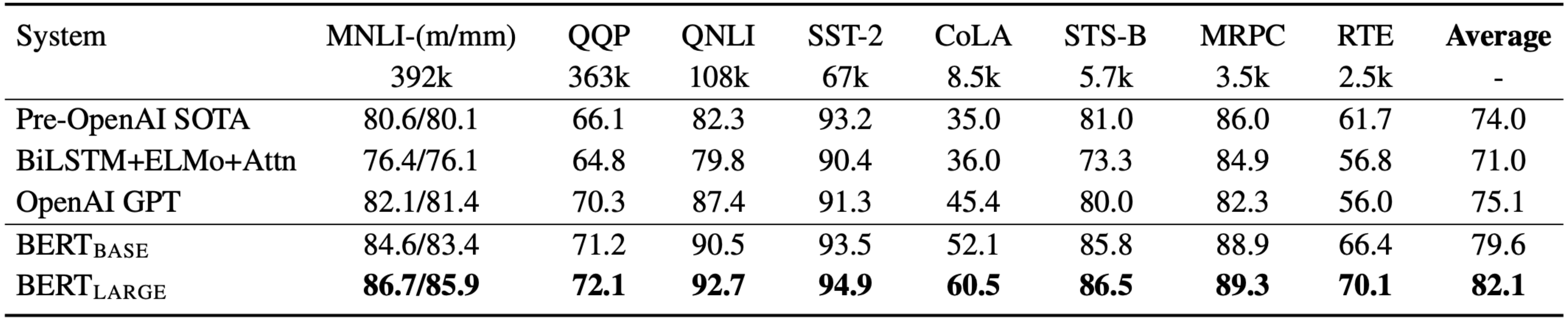
# GPT and GPT-2

- 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) | QQP | QNLI | SST-2 | CoLA | STS-B | MRPC | RTE | **Average** |
|---|---|---|---|---|---|---|---|---|---|
| | 392k | 363k | 108k | 67k | 8.5k | 5.7k | 3.5k | 2.5k | - |
| 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 |
| BERT$_{\text{BASE}}$ | 84.6/83.4 | 71.2 | 90.5 | 93.5 | 52.1 | 85.8 | 88.9 | 66.4 | 79.6 |
| 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

# VAE

## Idea 1

$x \longmapsto \mathrm{Enc}_\varphi(x) = p_\varphi(Z|x)$ Gaussian ---> sample z's

$z \longmapsto \mathrm{Dec}_\theta(z) = p_\theta(X|z) = \left[p_\theta(z)\right]_{j=1:m}$ ← Bernoulli

↓

sample $\tilde{x}$'s

$p = \binom{\text{Prob}}{\text{density}}$

$x \in \{0,1\}^d$

$Z, X$ = random variable

$z, x$ = actual values

$x \longmapsto \mu_\varphi(x), \ln\sigma_\varphi(x)$ $\in \mathbb{R}^m$

$P_\varphi(Z|x)$ variational

$x \xrightarrow{\boxed{\varphi}} \mu(x), \ln\sigma(x) \longrightarrow z = \mu + \sigma\varepsilon \longmapsto \boxed{\theta} \; P_\theta(X|z)$

sample $\varepsilon \sim N(0, I_m)$

↓

sample $\tilde{x}$

$p(x)$ = likelihood of data point x

$p(X)$ = model for X distribution

$z \in \mathbb{R}^m$

$m < d$

## Step 2 "Likelihoods

$$P_\theta(X,Z) = P_\theta(Z) \cdot P_\theta(X|Z)$$

$\tilde{x}$

Gaussian $\approx N(0,I)$

decoder

$$\left[ P_\theta(X) = \int_{\mathbb{R}^m} p(X,z)\,dz \right]$$

$$P_\theta(X) = \frac{P_\theta(X,z)}{P_\theta(z|X)} \simeq P_\varphi$$

← Ex

Idea 3.1

# Idea 3.2 Variational approx

Max Likelihood

$x \in \mathcal{D}$

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

log-Likelihood

$$= \underbrace{E_\varphi\left[\ln p_\theta(x,Z) - \ln p_\varphi(Z|x)\right]}_{\text{ELBO}} + \underbrace{E_\varphi\left[\ln \frac{p_\varphi(Z|x)}{p_\theta(Z|x)}\right]}_{\geq 0}$$

$\uparrow$ $\max\limits_{\theta,\varphi}$

$\uparrow$ samples $z \sim p_\varphi$

ELBO = Evidence Lower Bound

$$\int p_\theta(x,Z)\,dz = \text{evidence}$$

Decoder

$$\max_\theta \ \operatorname{avg}_z \ \underbrace{\ln p_\theta(x|z)}_{} -$$

$$\operatorname{avg}_z \sum_j \left[x_j \ln p_j(z,\theta) + (1-x_j)\ln(1-p_j(z,\theta))\right]$$

$$\max_\varphi \ E_\varphi\left[\ln p_\theta(x|z) - \ln p_\varphi(Z|x)\right]$$

↳ additional "sampling trick" needed

# ELBO

$$E_{\varphi}\Big[\ln \underbrace{p_\theta(x,z)}_{p_\theta(x|z)\cdot \underbrace{p(z)}_{N(0,I)}} - \ln p_\varphi(z|x)\Big] = E_\varphi\big[\ln p_\theta(x|z)\big] - \underbrace{E_\varphi\big[\ln p_\varphi(z|x) - \ln N(0,I)\big]}_{KL(p_\varphi(z|x)\,\|\,N(0,I))}$$

decoder

to compute $\frac{\partial ELBO}{\partial \varphi}$

## Sampling trick

- Wanted: samples from a distribution that doesn't depend on $\varphi$
- Idea: 1) sample $\varepsilon \sim N(0, I_m) \rightarrow z = \mu_\varphi(x) + \sigma_\varphi(x)\,\varepsilon$ ---> $n_\varepsilon$ samples ⓓ

Ex $\boxed{dz = \operatorname{diag}\{\sigma(x)\}\, d\varepsilon}$

- 2) change of variable

ELBO

$$\boxed{\int\big[\ln p_\theta(x|z) - \ln p_\varphi(z|x) + \ln p(z)\big]\, p_\varphi(z|x)\, dz} \;*$$

ⓐ $\prod_{j=1}^{m} \sigma_j \, d\varepsilon_1 \cdots d\varepsilon_m$

$-\frac{\|z\|^2}{2} - ct$ ⓔ

ⓑ $p_\varphi(z) = \exp\Big\{-\sum_{j=1}^{m} \frac{(z_j-\mu_j)^2}{2\sigma_j^2}\Big\} \cdot \frac{1}{(2\pi)^{\frac{m}{2}} \prod_{j=1}^{m} \sigma_j}$

ⓒ $\ln p_\varphi(z) = -\sum_{j=1}^{m} \frac{(z_j-\mu_j)^2}{2\sigma_j^2} - \sum_j \ln \sigma_j - \text{const}$

see Note

$$** \approx \overset{ⓓ}{\frac{1}{n_\varepsilon}} \sum_\varepsilon \Big[\ln p_\theta\big(x\,|\,z(\varepsilon)\big) + \underbrace{\sum_{j=1}^{m} \frac{\big(z_j(\varepsilon_j) - \mu_j(x)\big)^2}{2\sigma_j(x)^2} + \sum_j \ln \sigma_j(x)}_{ⓒ} - \frac{\|z\|^2}{2}\Big]$$

Now OK to take $\frac{\partial}{\partial \varphi}$ $\frac{\partial}{\partial \theta}$

$z\,|\,x, \varepsilon, \varphi$

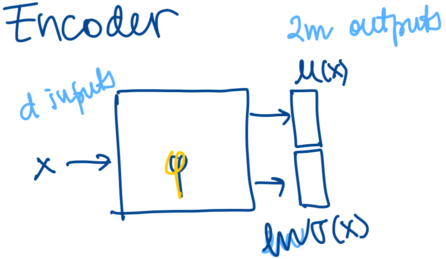
$$\sigma_\varphi(x) = \begin{bmatrix}\sigma_1 \\ \vdots \\ \sigma_m\end{bmatrix} \quad \mu_\varphi(x) = \begin{bmatrix}\mu_1 \\ \vdots \\ \mu_m\end{bmatrix} \quad \varepsilon = \begin{bmatrix}\varepsilon_1 \\ \vdots \\ \varepsilon_m\end{bmatrix}, \ldots$$

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

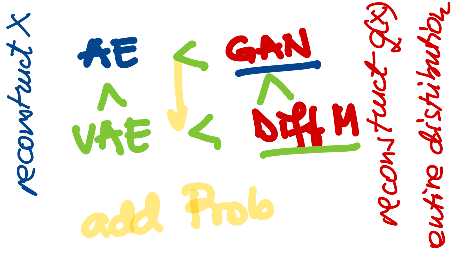
c $\dfrac{z_j - \mu_j}{\sigma_j} = \varepsilon_j \Rightarrow$ indep of $\sigma_\varphi, \mu_\varphi$!

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

Encoder



# Generative Models

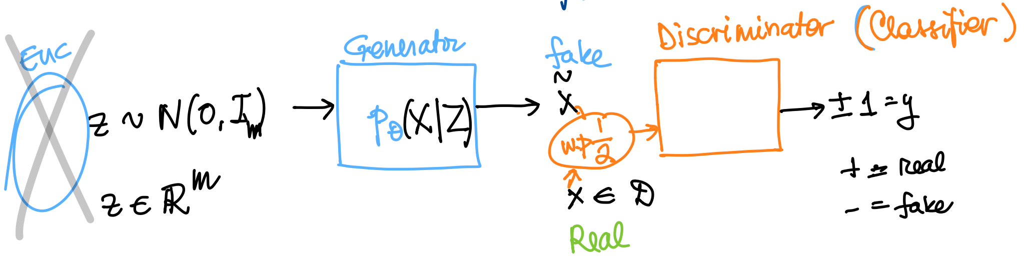


Generative Adversarial Networks (GAN)

Goal : given $\mathcal{D} = \{x^1, \dots x^n\} \sim q \leftarrow$ true

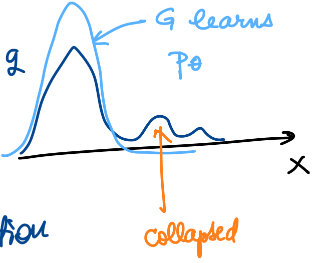
want $q \approx p_\theta$ and new samples

$\tilde{x} \sim p_\theta$



Problems

- mode collapse → G learns well only some modes of q
- "vanishing grads for G" → D learns too fast ⇒ always discriminates ⇒ G never "wins" ⇒ which direction to improve ?



# Diffusion Models

true $\mathcal{D}$ data set

$$q \sim X \equiv X^{(0)} \to \cdots \boxed{\phantom{x}} \to \boxed{\phantom{x}} \to \quad X^{(T)} = z \sim N(0, I_d) \qquad z \in \mathbb{R}^d$$

$\mathbb{R}^d$ $\quad x^{(t)} \quad x^{(t+1)}$ $\quad$ output

$t = 0{:}T$ Layers

F: $\quad q(x^{(t+1)} \mid x^{(t)}) = N(\alpha_t x^{(t)}, \beta_t I)$

$$x^{(t+1)} = \sqrt{\alpha_t}\, x^{(t)} + \sqrt{\beta_t}\, z^{(t)} \leftarrow N(0, I_d)$$

$$[\alpha_t + \beta_t = 1] \Rightarrow \mathrm{Var}\, x^{(t)} = I_d \quad \forall t$$

B: $\quad \underset{p^{(0)}}{\overset{\approx}{\tilde{x}}} \longleftarrow \qquad \leftarrow x^{(t)} \leftarrow x^{t+1} \leftarrow \quad \leftarrow x^{(T)} \sim N(0, I)$

$$p_\theta(x^{(t)} \mid x^{(t+1)}) = N\left(\mu_t(x^{(t+1)}) \mid \sigma_t^2(x^{(t+1)})\right)$$

$$\boxed{\begin{matrix}\mu \\ \mu\,\sigma\end{matrix}} \leftarrow \boxed{\theta_t} \leftarrow x^{(t+1)}$$

$$p(x^{(t+1)} \mid x^{(t)})$$