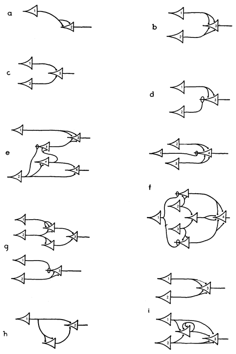
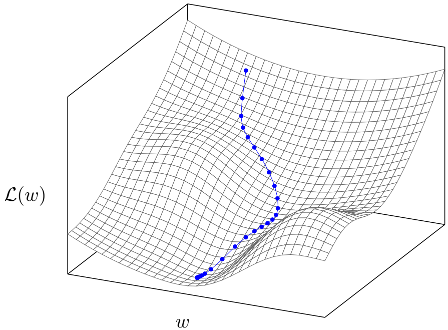
The Free Transformer

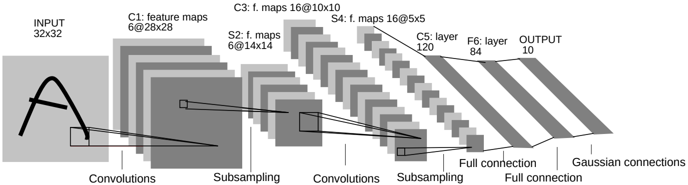
François Fleuret

Meta

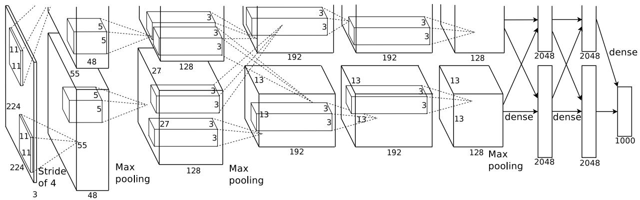


(McCulloch and Pitts, 1943)

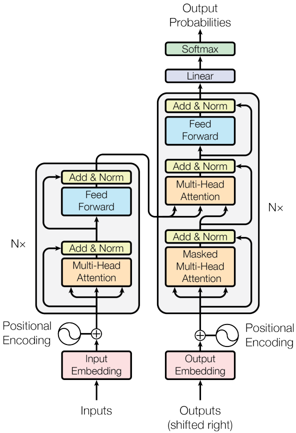




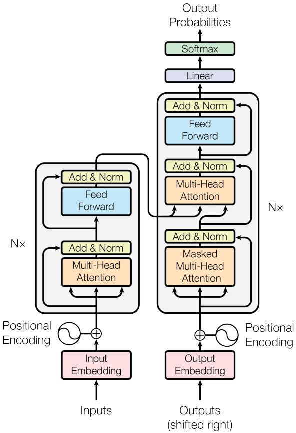
Convolutions (LeNet, 1989)



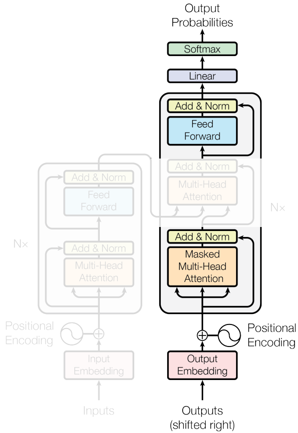
Very large models + GPUs (AlexNet, 2012)



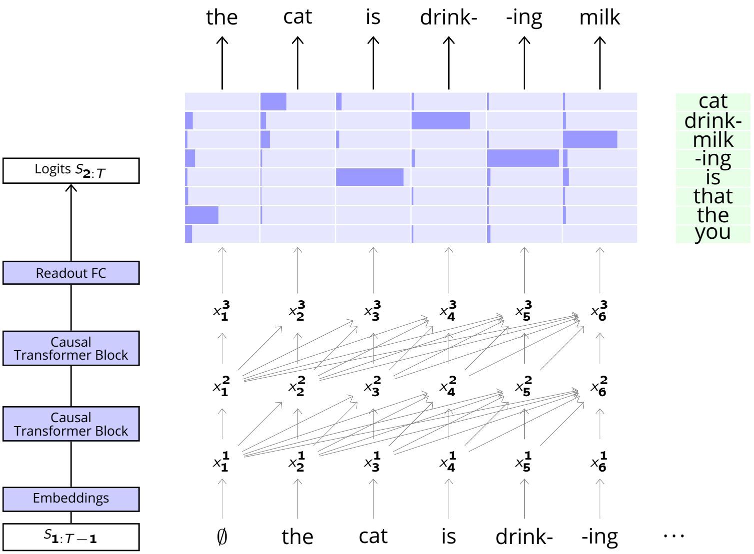
Transformer
(Vaswani et al., 2017)



Transformer
(Vaswani et al., 2017)



Decoder-only Transformer
(Radford et al., 2018)



the
cat
is
drink-
-ing
milk
cat
drink-
milk
-ing
is
that
the
you
Logits $S_{2:T}$
Readout FC
Causal Transformer Block
Causal Transformer Block
Embeddings
$S_{1:T-1}$
x_1^3 x_2^3 x_3^3 x_4^3 x_5^3 x_6^3
x_1^2 x_2^2 x_3^2 x_4^2 x_5^2 x_6^2
x_1^1 x_2^1 x_3^1 x_4^1 x_5^1 x_6^1
$\emptyset$
the
cat
is
drink-
-ing
$\cdots$

Such a model, trained only to generate text, can already be put to use for instance for classification.

```
I: water boils at 100 degrees, O: physics.  I: the square root of two is
irrational, O: mathematics.  I: the set of prime numbers is infinite, O:
mathematics.  I: gravity is proportional to the mass, O:
```

Such a model, trained only to generate text, can already be put to use for instance for classification.

```
I: water boils at 100 degrees, O: physics.  I: the square root of two is
irrational, O: mathematics.  I: the set of prime numbers is infinite, O:
mathematics.  I: gravity is proportional to the mass, O: physics.
```

Such a model, trained only to generate text, can already be put to use for instance for classification.

```
I: water boils at 100 degrees, O: physics.  I: the square root of two is
irrational, O: mathematics.  I: the set of prime numbers is infinite, O:
mathematics.  I: gravity is proportional to the mass, O: physics.
```

```
I: water boils at 100 degrees, O: physics.  I: the square root of two is
irrational, O: mathematics.  I: the set of prime numbers is infinite, O:
mathematics.  I: squares are rectangles, O:
```

Such a model, trained only to generate text, can already be put to use for instance for classification.

```
I: water boils at 100 degrees, O: physics.  I: the square root of two is
irrational, O: mathematics.  I: the set of prime numbers is infinite, O:
mathematics.  I: gravity is proportional to the mass, O: physics.
```

```
I: water boils at 100 degrees, O: physics.  I: the square root of two is
irrational, O: mathematics.  I: the set of prime numbers is infinite, O:
mathematics.  I: squares are rectangles, O: mathematics.
```

Such a model, trained only to generate text, can already be put to use for instance for classification.

```
I: water boils at 100 degrees, O: physics.  I: the square root of two is
irrational, O: mathematics.  I: the set of prime numbers is infinite, O:
mathematics.  I: gravity is proportional to the mass, O: physics.
```

```
I: water boils at 100 degrees, O: physics.  I: the square root of two is
irrational, O: mathematics.  I: the set of prime numbers is infinite, O:
mathematics.  I: squares are rectangles, O: mathematics.
```

```
I: I love apples, O: positive.  I: music is my passion, O: positive.  I: my
job is boring, O: negative.  I: frozen pizzas are awesome, O:
```

Such a model, trained only to generate text, can already be put to use for instance for classification.

`I: water boils at 100 degrees, O: physics. I: the square root of two is irrational, O: mathematics. I: the set of prime numbers is infinite, O: mathematics. I: gravity is proportional to the mass, O: physics.`

`I: water boils at 100 degrees, O: physics. I: the square root of two is irrational, O: mathematics. I: the set of prime numbers is infinite, O: mathematics. I: squares are rectangles, O: mathematics.`

`I: I love apples, O: positive. I: music is my passion, O: positive. I: my job is boring, O: negative. I: frozen pizzas are awesome, O: positive.`

Such a model, trained only to generate text, can already be put to use for instance for classification.

```
I: water boils at 100 degrees, O: physics.  I: the square root of two is
irrational, O: mathematics.  I: the set of prime numbers is infinite, O:
mathematics.  I: gravity is proportional to the mass, O: physics.
```

```
I: water boils at 100 degrees, O: physics.  I: the square root of two is
irrational, O: mathematics.  I: the set of prime numbers is infinite, O:
mathematics.  I: squares are rectangles, O: mathematics.
```

```
I: I love apples, O: positive.  I: music is my passion, O: positive.  I: my
job is boring, O: negative.  I: frozen pizzas are awesome, O: positive.
```

```
I: I love apples, O: positive.  I: music is my passion, O: positive.  I:
my job is boring, O: negative.  I: frozen pizzas taste like cardboard, O:
```

Such a model, trained only to generate text, can already be put to use for instance for classification.

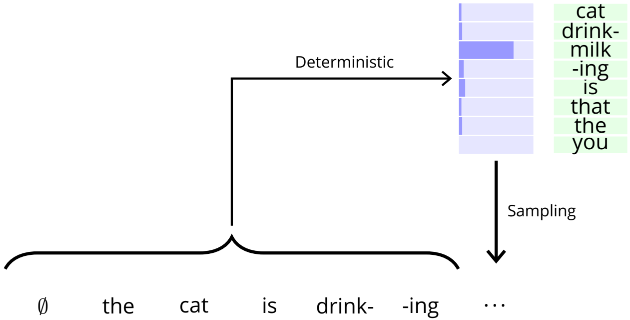
```
I: water boils at 100 degrees, O: physics.  I: the square root of two is
irrational, O: mathematics.  I: the set of prime numbers is infinite, O:
mathematics.  I: gravity is proportional to the mass, O: physics.
```

```
I: water boils at 100 degrees, O: physics.  I: the square root of two is
irrational, O: mathematics.  I: the set of prime numbers is infinite, O:
mathematics.  I: squares are rectangles, O: mathematics.
```

```
I: I love apples, O: positive.  I: music is my passion, O: positive.  I: my
job is boring, O: negative.  I: frozen pizzas are awesome, O: positive.
```

```
I: I love apples, O: positive.  I: music is my passion, O: positive.  I:
my job is boring, O: negative.  I: frozen pizzas taste like cardboard, O:
negative.
```

The problem with autoregression



The only sampling in a Transformer are the tokens.

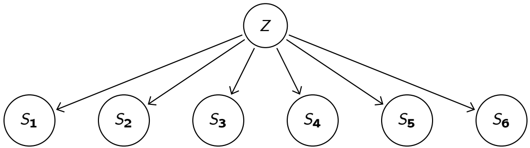
Consider a corpus of positive and negative online reviews.

$$Z \sim \mathcal{U}\left(\{-1, 1\}\right), \ S \sim \mu_Z.$$

An AR model has no way of implementing this factorized distribution, it cannot “decide” beforehand what type of review to generate.

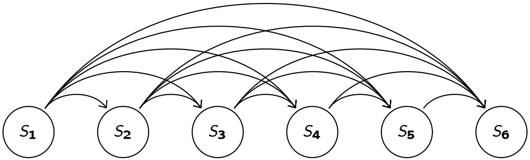
It would estimate on the fly the negativity / positivity of what it has written so far to be consistent.

The “true model” of a joint distribution, with latents, is usually simpler and more robust than the “latentless” AR model.



$$P(S_{t+1} = 1 \mid Z, S_1, \ldots, S_t) = (1-\epsilon)Z + \epsilon(1-Z)$$

The “true model” of a joint distribution, with latents, is usually simpler and more robust than the “latentless” AR model.



$$\mathrm{P}(X_{t+1} = 1 \mid X_1 = x_1, \ldots, X_t = x_t) =$$

$$\frac{\left(\frac{\epsilon}{1-\epsilon}\right)^{\sum_{s=1}^{t} x_s} (1-\epsilon)^t \epsilon + \left(\frac{1-\epsilon}{\epsilon}\right)^{\sum_{s=1}^{t} x_s} \epsilon^t (1-\epsilon)}{\left(\frac{\epsilon}{1-\epsilon}\right)^{\sum_{s=1}^{t} x_s} (1-\epsilon)^t + \left(\frac{1-\epsilon}{\epsilon}\right)^{\sum_{s=1}^{t} x_s} \epsilon^t}.$$

We could prefix every training sample with a token Z indicating if the review is positive or not.

We could prefix every training sample with a token Z indicating if the review is positive or not.

Reasoning post-training tries to address it unsupervised, however:

+ it requires a trained model,
+ current approaches cast it as reinforcement learning,
+ the conditionning variable are discrete tokens,
+ it cannot deal with stochastic responses.

We propose instead to let the model build latent variables

$$Y_r = f_r(S_1, \dots, S_t, Y_1, \dots, Y_{r-1}, Z_r; \theta)$$

where Z_r is sampled from a random generator. This can be interpreted as making decisions beside the token choices.

We propose instead to let the model build latent variables

$$Y_r = f_r(S_1, \ldots, S_t, Y_1, \ldots, Y_{r-1}, Z_r; \theta)$$

where Z_r is sampled from a random generator. This can be interpreted as making decisions beside the token choices.

Sampling from such a trained model is trivial: sample z, and run the AR process as usual to sample $\mathrm{P}_\theta(S \mid Z = z)$.

Training, however, is far more involved.

The Variational Autoencoder

Given a training sample s, we want to maximize

$$\mathrm{P}_\theta(S = s) = \mathbb{E}_{z \sim \mathrm{P}(Z)}\Big[\mathrm{P}_\theta(S = s \mid Z = z)\Big]$$

Given a training sample s, we want to maximize

$$\mathrm{P}_\theta(S = s) = \mathbb{E}_{z \sim \mathrm{P}(Z)} \Big[\mathrm{P}_\theta(S = s \mid Z = z) \Big]$$

Since $\mathrm{P}_\theta(S = s \mid Z = z)$ is a full-fledged model, there is no closed form for $P_\theta(S = s)$, and numerical integration requires "good" Zs.

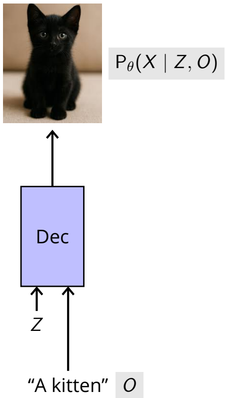
Given a training sample s, we want to maximize

$$\mathrm{P}_\theta(S = s) = \mathbb{E}_{z \sim \mathrm{P}(Z)} \Big[\mathrm{P}_\theta(S = s \mid Z = z) \Big]$$

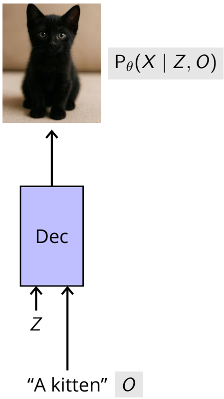
Since $\mathrm{P}_\theta(S = s \mid Z = z)$ is a full-fledged model, there is no closed form for $P_\theta(S = s)$, and numerical integration requires "good" Zs.

The Variational Autoencoder proposed by Kingma and Welling (2013) provides a formal derivation to train a sampler $q_\theta(Z; S = s)$.

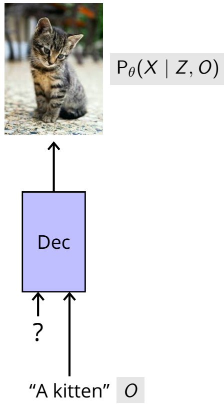
Inference



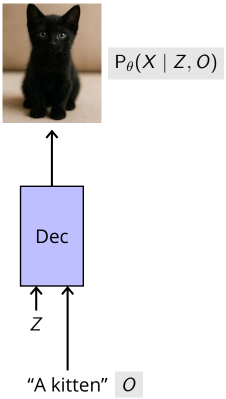
Inference



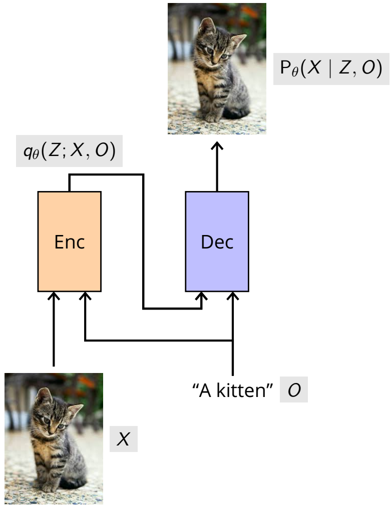
Training



Inference



Training



$$\log \mathrm{P}_\theta(S = s)$$

$$\log \mathrm{P}_\theta(S = s)$$

$$\geq \log \mathrm{P}_\theta(S = s) - \overbrace{\mathbb{D}_{\mathrm{KL}}\Big(q_\theta(Z; S = s) \,\Big\|\, \mathrm{P}_\theta(Z \mid S = s)\Big)}^{\text{How bad the sampler is}}$$

$$\log \mathrm{P}_\theta(S = s)$$

$$\geq \log \mathrm{P}_\theta(S = s) - \overbrace{\mathbb{D}_{\mathrm{KL}}\Big(q_\theta(Z; S = s) \Big\| \mathrm{P}_\theta(Z \mid S = s)\Big)}^{\text{How bad the sampler is}}$$

$$= \underbrace{\mathbb{E}_{z \sim q_\theta(Z; S=s)} \left[\log \frac{\mathrm{P}_\theta(S = s, Z = z)}{q_\theta(Z = z; S = s)}\right]}_{\text{"Evidence Lower Bound" (aka ELBO)}}$$

$$\log \mathrm{P}_\theta(S = s)$$

$$\geq \log \mathrm{P}_\theta(S = s) - \overbrace{\mathbb{D}_{\mathrm{KL}}\Big(q_\theta(Z; S = s) \,\Big\|\, \mathrm{P}_\theta(Z \mid S = s)\Big)}^{\text{How bad the sampler is}}$$

$$= \underbrace{\mathbb{E}_{z \sim q_\theta(Z; S=s)}\left[\log \frac{\mathrm{P}_\theta(S = s, Z = z)}{q_\theta(Z = z; S = s)}\right]}_{\text{“Evidence Lower Bound” (aka ELBO)}}$$

$$= \underbrace{\overbrace{\mathbb{E}_{z \sim q_\theta(Z; S=s)}\Big[\log \mathrm{P}_\theta(S = s \mid Z = z)\Big]}^{-\text{cross-entropy}} - \overbrace{\mathbb{D}_{\mathrm{KL}}\Big(q_\theta(Z; S = s) \,\Big\|\, \mathrm{P}(Z)\Big)}^{\text{How much we cheat}}}_{\text{The quantity we maximize during training}}$$

The model $q_\theta(Z = z; S = s)$ can be envisioned as an encoder that extracts the necessary information from s.

E.g. “s is a negative review, θ should be improved for $z = -1$”.

The model $q_\theta(Z = z; S = s)$ can be envisioned as an encoder that extracts the necessary information from s.

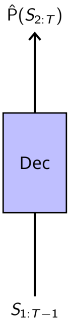
E.g. "s is a negative review, θ should be improved for $z = -1$".

The term

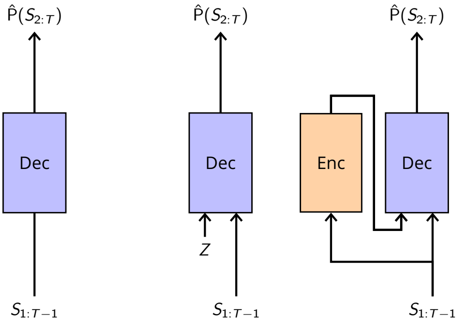
$$\mathbb{D}_{\mathsf{KL}}\left(q_\theta(Z; S = s) \,\middle\|\, \mathsf{P}(Z)\right)$$

reflects how much information about s the encoder provides to the decoder, and must be controlled carefully during training.

The Free Transformer

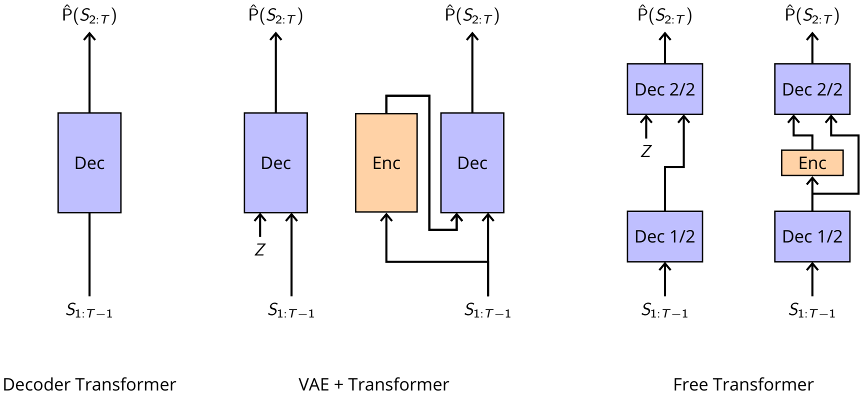


Decoder Transformer



Decoder Transformer

VAE + Transformer


$\hat{P}(S_{2:T})$
Dec
$S_{1:T-1}$
$\hat{P}(S_{2:T})$
Dec
Z
$S_{1:T-1}$
$\hat{P}(S_{2:T})$
Enc
Dec
$S_{1:T-1}$
$\hat{P}(S_{2:T})$
Dec 2/2
Z
Dec 1/2
$S_{1:T-1}$
$\hat{P}(S_{2:T})$
Dec 2/2
Enc
Dec 1/2
$S_{1:T-1}$
Decoder Transformer
VAE + Transformer
Free Transformer

We use for Z a one-hot vector of dimension 2^{16}, comparable to the vocabulary size of 2^{17}

We use for Z a one-hot vector of dimension 2^{16}, comparable to the vocabulary size of 2^{17}

Given $L \in \mathrm{R}^H$, the "Binary Mapper" interprets them as bits logits and

+ Computes the distribution p over $\{0, \dots, 2^H - 1\}$.
+ Samples $K \sim p$.
+ Outputs $Y_d = \delta_{d=K} + p_d - \mathrm{detach}(p_d)$.

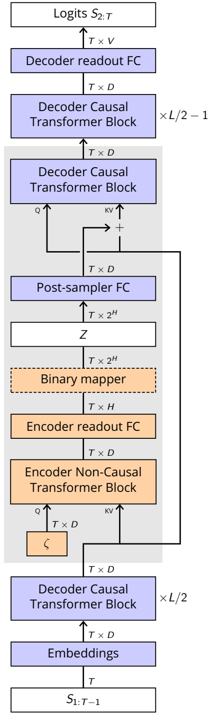
We use for Z a one-hot vector of dimension 2^{16}, comparable to the vocabulary size of 2^{17}

Given $L \in \mathrm{R}^H$, the "Binary Mapper" interprets them as bits logits and

+ Computes the distribution p over $\{0, \ldots, 2^H - 1\}$.
+ Samples $K \sim p$.
+ Outputs $Y_d = \delta_{d=K} + p_d - \mathrm{detach}(p_d)$.

The vector L is:

+ Constant zero for $P(Z)$.
+ The output of the encoder for $q_\theta(Z; S = s)$.



Logits $S_{2:T}$
$T \times V$
Decoder readout FC
$T \times D$
Decoder Causal Transformer Block
$\times L/2 - 1$
$T \times D$
Decoder Causal Transformer Block
Q
KV
+
$T \times D$
Post-sampler FC
$T \times 2^H$
Z
$T \times 2^H$
Binary mapper
$T \times H$
Encoder readout FC
$T \times D$
Encoder Non-Causal Transformer Block
Q
KV
$T \times D$
ζ
$T \times D$
Decoder Causal Transformer Block
$\times L/2$
$T \times D$
Embeddings
T
$S_{1:T-1}$

The KL divergence can be expressed per token

$$\mathbb{D}_{\mathsf{KL}}\left(q(Z_t; S_1, \dots, S_T) \,\middle\|\, \mathsf{P}(Z_t)\right) = H \log 2 + \sum_{z=1}^{2^H} q(Z_t = z; S) \log q(Z_t = z; S).$$

The KL divergence can be expressed per token

$$\mathbb{D}_{\mathsf{KL}}\left(q(Z_t; S_1, \dots, S_T) \,\middle\|\, \mathsf{P}(Z_t)\right) = H \log 2 + \sum_{z=1}^{2^H} q(Z_t = z; S) \log q(Z_t = z; S).$$

We use the free-bits method (Kingma et al., 2016), also per token

$$\mathcal{L}_{\mathsf{KL}} = \frac{1}{T} \sum_{t=1}^{T} \max\left(0,\ \mathbb{D}_{\mathsf{KL}}\left(q(Z_t; S_1, \dots, S_T) \,\middle\|\, \mathsf{P}(Z_t)\right) - \kappa\right).$$

Experiments

```
K>!_________!_______!_______!____________!_______KKKKKKKK_________
C>___CCCCCCCC_________________!______!_____________!__!__!________
X>___________________!!_________!!___XX!XXXXX_____!_______________
R>!__RRRRRRRR_____________!__!_______________________________!____
P>!__!___________________________________________________PPPPPPPP_
L>_______!_!LLLLLLLL________!___________!!____________!___________
V>__!_________________!__!________VVVVVV!V________!_____!____!____
P>_________PPPPPPPP_____!________________!_______________________!
A>_______!___________!___________________________!_______AAAAAAA!_
P>____________________!____PPPPPP!P____!___________!_________!__!!
I>__________________________________________!_!__IIIIIIII_________
D>______!_!___________________________!_________!DDDDDDD__________
A>_____!___AAAAAAA!_______________!_________________!______!______
J>_______!_____!_________J!JJJJJJ_____________!___________________
```

Training Examples

```
T>_________________________________TTTTTTTT_______________________
T>________________________________TTTTTTTT________________________
T>_________________________________TTTTTTTT_______________________
T>_____________________________________!TTTTTTTT__________________
T>_____________________________________________!_______TTTTTTTT___
```

```
T>_________________________________TTTTTTTT_______________________
T>_____________________________________________________TTTTTTTT___
T>______________________________________________________!TTTTTTTT_
T>___________________________!____________________________TTTTTTTT
T>_______________________________________________________TTTTTTTT_
```

```
T>_____________________________________________________TTTTTTTT___
T>_____________________________________________________TTTTTTTT___
T>___________________________________TTTTTTTT______!_____!____!___
T>_____________________________________________TTTTTTTT___________
T>__________________________________TTTTTTTT______________________
```

$$\kappa = \log(2)/64$$

```
F>_______________________________________________________FFFFFFFF_
F>___________________FFFFFFFF__________!__________!_______________
F>_________________FFFFFFFF________________________!____________!_
F>___________________________________FFFFFFFF__________________!__
F>____________________________________________!FFF!FFFF___________
```

```
F>_______________________!____________________________FFFFFFFF____
F>____________________________________________________FFFFFFFF____
F>____________________________________________________FFFFFFFF!___
F>___________________________________________________FFFFFFFF!____
F>_____________________________________________________FFFFFFFF___
```

```
F>_________________________FFFFFFFF!_________________!____________
F>__________!____________FF!FFFFF_________________________________
F>________________________FFFFFFFF___________________!____________
F>_______________________FFFFFFFF_______________!______!__________
F>_______________________FFFFFFFF______________________!__________
```

$$\kappa = \log(2)/8$$

```
J>JJJJJJJ!____!_________!!__!_!_!__________!___________!___!__!___
J>_____!_____!______!______!_JJJJJJJJ______________________!______
J>___JJJ!JJJJ____________!__!___!_!__!_____!_____!!__!_____!___!__
J>__!___________JJJJJJJJ___________________!________!____!________
J>______!!___!!!_____________JJJJJJJJ!______!!!_!_!___!___________
```

```
J>_________JJJ!JJJJ__!______________!__________!!___!!_________!__
J>_________JJJ!JJJJ__!______________!______!____!__!!__________!__
J>_________JJ!JJJJJ__!_______!________!________!!__!!__________!__
J>_________JJJ!JJJJ__!________________!!____!!_!!____!_________!__
J>___!_____JJ!JJJJJ__!______________!_______!__!!_______!______!__
```

```
J>__JJJJJJJJ______!___________!____!_!______!_______!__!________!_
J>__JJJJJJJJ______!___________!____!_!________!__!__!__!________!_
J>_JJJJJJJJ_______!_______!________!_!______!_______!__!_______!__
J>_JJJJJJJJ_______!________________!_!________!__!____!!___!____!_
J>_JJJJJJJJ_______!___________!_____!_!_____!_______!__!_______!__
```

$$\kappa = \log(2)$$

```
O>___________!!________!__!________________!_____!_______!______!!
O>____OOOOO______________________________________________________!
O>O!___O_!__!_!__!__!_OO____!!__OO_________________!!________!___!
O>_____!_____!_____________!_______________!_F___!!_!_______!_____
O>__OOOO!OO!___OO____!____O_!________________________O!____O_____!
```

```
O>_________OOOO________O__________!_____________________!!_!O____!
O>_________OO!O________O____O_____!_____________________!!!!O____!
O>_________OO!O________O____O_____!________________O____!__!O____!
O>_________OOOO________O____O_____!_____________________!!_!O____!
O>_________OOOO________O____O_____!______________________!_!O____!
```

```
O>__!___OO______________!___OO!O________O!O____________O_____O___!
O>O_!__OOO__________________OOOO________O!O____________O________!!
O>__!___OO__________________OO__________O!O____________O_____O____
O>__!___OO__________________OO__________O!O____O_______O_____O__!!
O>O_!__OOO_________!OO__!___OO__________O!O____________O_____O___!
```

$$\kappa = 8\log(2)$$

We benchmark the Free Transformer with the following configurations:

+ 1.5B model (Qwen like)
    - 28 layers, embedding / readout weight tying
    - 1536 model dimensions
    - GQA, 12 query heads, 2 key-value heads
    - 47B tokens (32 H100s for $\approx$ 12 hours)

+ 8B model (Llama-3 like)
    - 32 layers
    - 4096 model dimensions
    - GQA, 32 query heads, 8 key-value heads
    - 200B tokens (256 H100s for $\approx$ 24 hours), or with 1T tokens (5 days).

We benchmark the Free Transformer with the following configurations:

+ 1.5B model (Qwen like)
  - 28 layers, embedding / readout weight tying
  - 1536 model dimensions
  - GQA, 12 query heads, 2 key-value heads
  - 47B tokens (32 H100s for $\approx$ 12 hours)

+ 8B model (Llama-3 like)
  - 32 layers
  - 4096 model dimensions
  - GQA, 32 query heads, 8 key-value heads
  - 200B tokens (256 H100s for $\approx$ 24 hours), or with 1T tokens (5 days).

The dimension for Z is $2^{16} = 65536$, and the encoder transformer block requires $\approx$ 3% more compute.

1.5B models (47B tokens)

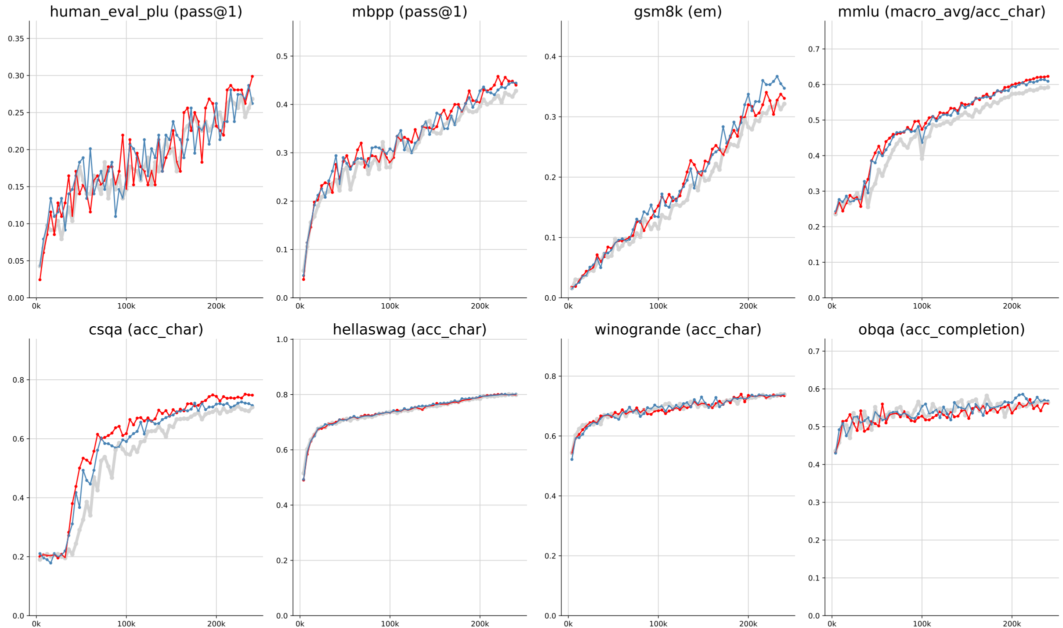
	Baseline	Free Transformer							
		1/4 bit		1/2 bit		1 bit		2 bits	
Generative code/math									
human_eval_plu (pass@1)	0.055	0.079	+44.44%	0.079	+44.44%	0.085	+55.56%	0.085	+55.56%
mbpp (pass@1)	0.112	0.144	+28.57%	0.148	+32.14%	0.152	+35.71%	0.122	+8.93%
gsm8k (em)	0.025	0.028	+12.12%	0.027	+6.06%	0.033	+30.30%	0.027	+6.06%
Multi-choice general knowledge / common sense									
mmlu (macro_avg/acc_char)	0.252	0.265	+5.31%	0.261	+3.76%	0.254	+1.07%	0.257	+2.19%
csqa (acc_char)	0.199	0.175	-11.93%	0.199	+0.00%	0.187	-6.17%	0.197	-0.82%
hellaswag (acc_char)	0.593	0.591	-0.40%	0.594	+0.15%	0.592	-0.27%	0.595	+0.32%
winogrande (acc_char)	0.603	0.604	+0.13%	0.598	-0.79%	0.600	-0.52%	0.597	-1.05%
obqa (acc_completion)	0.446	0.450	+0.90%	0.468	+4.93%	0.460	+3.14%	0.490	+9.87%
arc_challenge (acc_completion)	0.400	0.392	-1.93%	0.386	-3.43%	0.405	+1.29%	0.385	-3.65%
arc_easy (acc_completion)	0.596	0.602	+0.92%	0.592	-0.64%	0.603	+1.06%	0.592	-0.71%
piqa (acc_char)	0.734	0.736	+0.22%	0.738	+0.52%	0.734	+0.07%	0.733	-0.15%
Multi-choice text understanding									
race.high (acc_char)	0.390	0.382	-2.20%	0.390	+0.00%	0.387	-0.81%	0.386	-1.03%
race.middle (acc_char)	0.532	0.511	-3.93%	0.519	-2.49%	0.522	-1.83%	0.514	-3.40%
boolq (acc_completion)	0.583	0.632	+8.39%	0.614	+5.35%	0.648	+11.12%	0.620	+6.29%
Culture									
nq (em)	0.081	0.069	-15.36%	0.073	-9.56%	0.075	-7.17%	0.071	-11.95%
tqa (em)	0.205	0.191	-6.93%	0.190	-7.58%	0.200	-2.84%	0.197	-4.13%

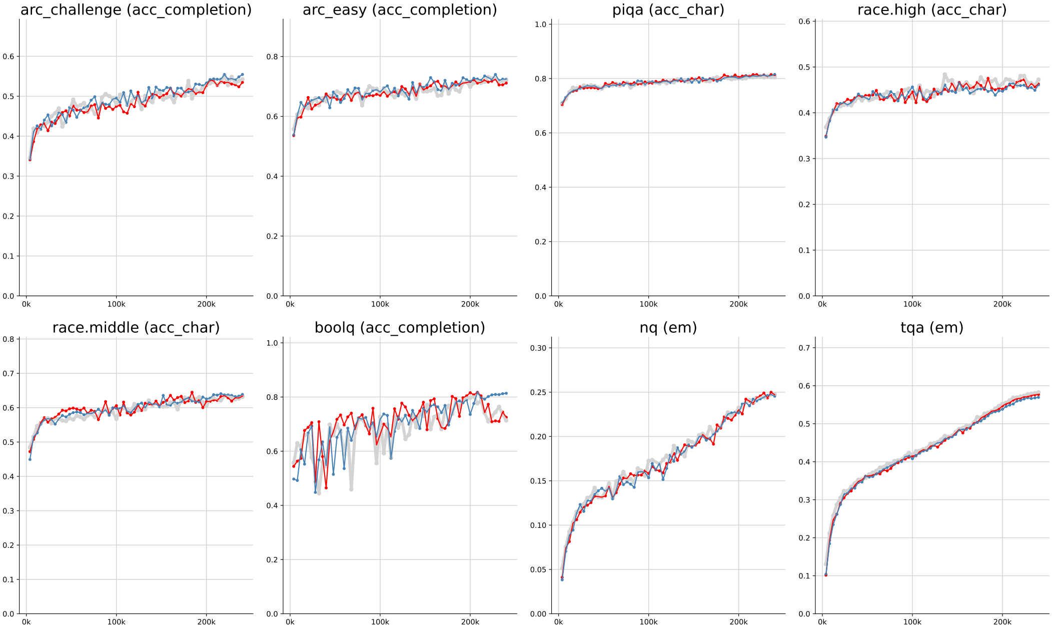
8B models (200B tokens)

	Baseline	**Free Transformer**							
		1/4 bit		1/2 bit		1 bit		2 bits	
Generative code/math									
human_eval_plu (pass@1)	0.159	0.171	+7.69%	0.189	+19.23%	0.165	+3.85%	0.177	+11.54%
mbpp (pass@1)	0.278	0.330	+18.71%	0.306	+10.07%	0.298	+7.19%	0.318	+14.39%
gsm8k (em)	0.086	0.079	-8.77%	0.095	+9.65%	0.104	+20.18%	0.096	+10.53%
Multi-choice general knowledge / common sense									
mmlu (macro_avg/acc_char)	0.359	0.337	-6.13%	0.398	+10.97%	0.365	+1.81%	0.345	-4.00%
csqa (acc_char)	0.356	0.292	-17.93%	0.450	+26.21%	0.346	-2.99%	0.324	-8.97%
hellaswag (acc_char)	0.735	0.737	+0.26%	0.737	+0.26%	0.732	-0.45%	0.738	+0.39%
winogrande (acc_char)	0.680	0.667	-1.86%	0.664	-2.32%	0.664	-2.32%	0.667	-1.86%
obqa (acc_completion)	0.522	0.508	-2.68%	0.484	-7.28%	0.530	+1.53%	0.554	+6.13%
arc_challenge (acc_completion)	0.465	0.483	+3.87%	0.468	+0.55%	0.452	-2.95%	0.485	+4.24%
arc_easy (acc_completion)	0.677	0.676	-0.25%	0.665	-1.81%	0.668	-1.44%	0.679	+0.31%
piqa (acc_char)	0.774	0.780	+0.77%	0.782	+1.05%	0.785	+1.41%	0.793	+2.46%
Multi-choice text understanding									
race.high (acc_char)	0.433	0.447	+3.30%	0.443	+2.25%	0.444	+2.58%	0.435	+0.53%
race.middle (acc_char)	0.594	0.592	-0.35%	0.591	-0.47%	0.587	-1.17%	0.584	-1.64%
boolq (acc_completion)	0.705	0.632	-10.37%	0.632	-10.33%	0.687	-2.47%	0.671	-4.82%
Culture									
nq (em)	0.181	0.183	+1.38%	0.167	-7.67%	0.173	-4.14%	0.168	-6.90%
tqa (em)	0.440	0.438	-0.28%	0.443	+0.80%	0.434	-1.19%	0.446	+1.45%

8B models (1T tokens)

	Final value					Average (last third)				
	Baseline	**Free Transformer**				**Baseline**	**Free Transformer**			
		1/2 bit		1 bit			1/2 bit		1 bit	
Generative code/math										
human_eval_plu (pass@1)	0.268	0.299	+11.36%	0.262	-2.27%	0.245	0.256	+4.22%	0.241	-1.74%
mbpp (pass@1)	0.428	0.440	+2.80%	0.444	+3.74%	0.396	0.421	+6.08%	0.412	+4.04%
gsm8k (em)	0.321	0.331	+2.83%	0.347	+8.02%	0.280	0.296	+5.84%	0.313	+11.96%
Multi-choice general knowledge / common sense										
mmlu (macro_avg/acc_char)	0.592	0.623	+5.20%	0.609	+2.88%	0.567	0.596	+5.16%	0.590	+4.19%
csqa (acc_char)	0.707	0.748	+5.79%	0.713	+0.81%	0.689	0.733	+6.28%	0.711	+3.18%
hellaswag (acc_char)	0.799	0.799	-0.01%	0.801	+0.30%	0.787	0.788	+0.18%	0.790	+0.37%
winogrande (acc_char)	0.739	0.735	-0.53%	0.740	+0.11%	0.725	0.727	+0.27%	0.728	+0.33%
obqa (acc_completion)	0.564	0.562	-0.35%	0.568	+0.71%	0.556	0.551	-0.86%	0.563	+1.33%
arc_challenge (acc_completion)	0.542	0.535	-1.42%	0.555	+2.22%	0.524	0.522	-0.40%	0.532	+1.57%
arc_easy (acc_completion)	0.721	0.711	-1.41%	0.724	+0.35%	0.706	0.711	+0.68%	0.717	+1.55%
piqa (acc_char)	0.805	0.812	+0.88%	0.815	+1.28%	0.802	0.807	+0.61%	0.804	+0.30%
Multi-choice text understanding										
race.high (acc_char)	0.473	0.463	-2.06%	0.461	-2.42%	0.467	0.460	-1.55%	0.453	-3.00%
race.middle (acc_char)	0.632	0.634	+0.33%	0.639	+1.10%	0.623	0.624	+0.16%	0.628	+0.80%
boolq (acc_completion)	0.713	0.725	+1.63%	0.814	+14.06%	0.755	0.754	-0.10%	0.781	+3.42%
Culture										
nq (em)	0.248	0.247	-0.22%	0.245	-0.89%	0.229	0.227	-0.76%	0.228	-0.63%
tqa (em)	0.583	0.577	-1.00%	0.569	-2.28%	0.549	0.544	-0.90%	0.539	-1.85%


human_eval_plu (pass@1)
mbpp (pass@1)
gsm8k (em)
mmlu (macro_avg/acc_char)
csqa (acc_char)
hellaswag (acc_char)
winogrande (acc_char)
obqa (acc_completion)
0k
100k
200k


arc_challenge (acc_completion)
arc_easy (acc_completion)
piqa (acc_char)
race.high (acc_char)
race.middle (acc_char)
boolq (acc_completion)
nq (em)
tqa (em)
0k
100k
200k

Conclusion

+ Seems to work.
+ In hindsight pretty obvious.
+ Lots of arbitrary design choices to ablate / improve.
+ May need a tailored optimization scheme.

+ Seems to work.
+ In hindsight pretty obvious.
+ Lots of arbitrary design choices to ablate / improve.
+ May need a tailored optimization scheme.

+ Unclear how it will hold on [very] large scale.
+ May have strengths not visible in current benchmarks.
+ Qualitative differences may justify more compute.

https://arxiv.org/abs/2510.17558

fleuret@meta.com

Questions?

D. P. Kingma and M. Welling. **Auto-Encoding Variational Bayes**, Dec. 2013. arXiv:1312.6114 [stat].

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. **Improving Variational Inference with Inverse Autoregressive Flow**, Jan. 2016. arXiv:1606.04934 [cs].

W. S. McCulloch and W. Pitts. **A logical calculus of the ideas immanent in nervous activity**. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. **Improving language understanding by generative pre-training**. Technical report, OpenAI, 2018.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. **Attention Is All You Need**, Aug. 2017. arXiv:1706.03762 [cs].