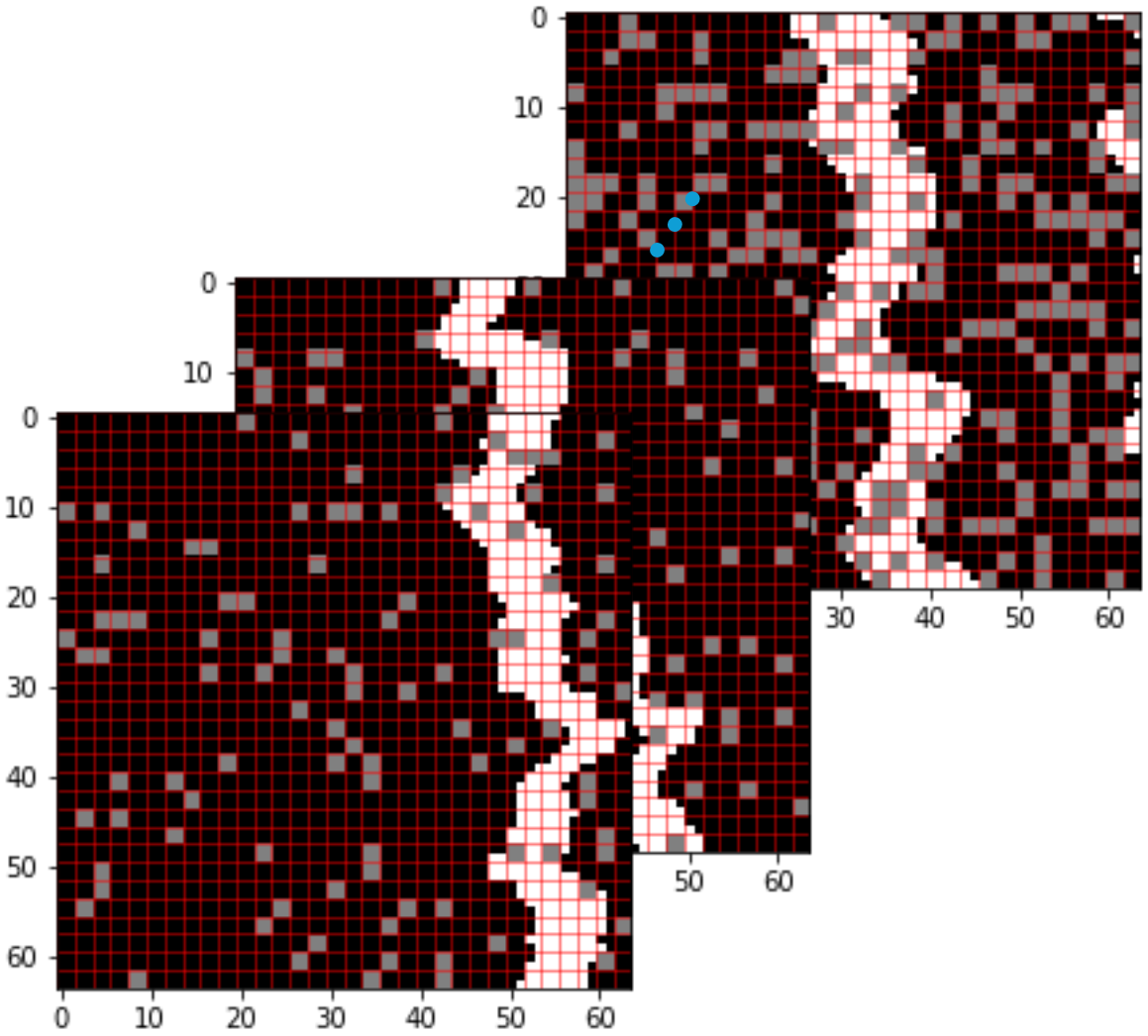
Transformer arkitektur

I detalj


FORWARD



Steg 1:

Input embeddings X (batch_size, num_patches, embed_dim)

batch_size: batch størrelse = 32

num_patches: Antall patcher = 1024 (32*32)

embed_dim: Embedding dimensjoner = 4

$$X[0,:,:] = \overset{1024}{\overset{\overset{4}{\downarrow}}{\begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0.5 & \cdots & 0.5 \end{pmatrix}}} \begin{matrix} \text{unmasked} \\ \\ \text{masked} \end{matrix}$$

Trenbare parametere

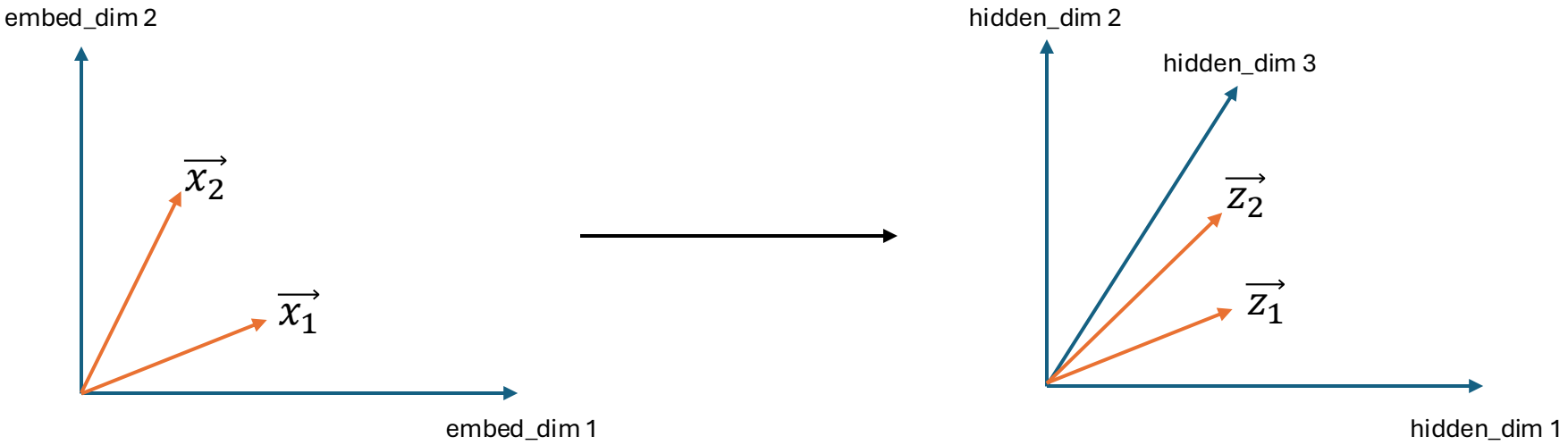
Steg 2:

$$Z = XA_{up}^{T} + b_{up}$$

NB!!! b_{up} har ikke samme dimensjon som P

- A_{up}(hidden_dim,embed_dim)
  - hidden_dim: antall dimensjoner i oppskalert embedding rom = 128
  - embed_dim: antall dimensjoner i originalt embedding rom = 4
- b_{up}(hidden_dim)

Vi oppskalerer input embeddings til et høy-dimensjonalt rom via en linær transformasjon.

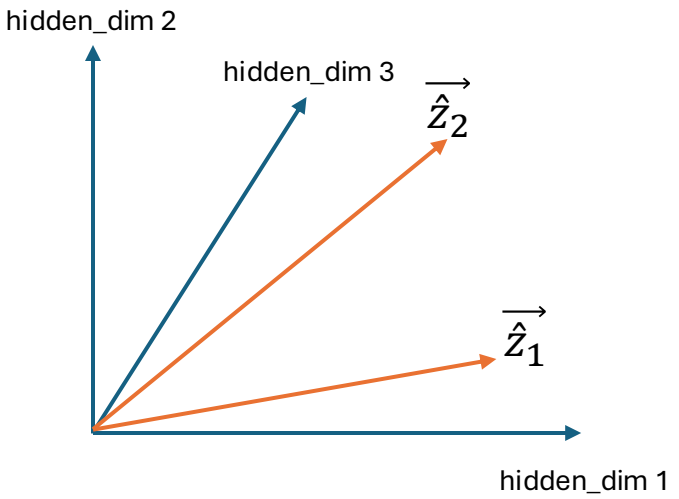


Steg 3:

$$\hat{Z} = Z + P$$

- $\hat{Z}$(batch_size,num_patches,hidden_dim)=(32,1024,128)
- Z(batch_size,num_patches,hidden_dim)=(32,1024,128)
- P(num_patches,hidden_dim)=(1024,128)

Vi legger til en Positional embedding matrix med 1024*128 trenbare parametere. Den lærer seg romlige sammenhenger mellom vector embeddings i det 128-dimensjonale rommet.



NB!!! Sa feil i møtet, vektmatrisene har dimensjon (hidden_dim,hidden_dim), IKKE (num_patches,hidden_dim).

Steg 4:

$$Q = W_q\hat{Z} \qquad \mathrm{K} = \mathrm{W_k}\hat{Z} \qquad V = W_v\hat{Z}$$

- Q(batch_size,num_patches,hidden_dim)=(32,1024,128)
- K(batch_size,num_patches,hidden_dim)=(32,1024,128)
- V(batch_size,num_patches,hidden_dim)=(32,1024,128)
- W_q(hidden_dim,hidden_dim)=(128,128)
- W_q(hidden_dim,hidden_dim)=(128,128)
- W_q(hidden_dim,hidden_dim)=(128,128)

Her regner vi ut query, key og value matrisene. De har samme dimensjon som input embedding ($\hat{Z}$).

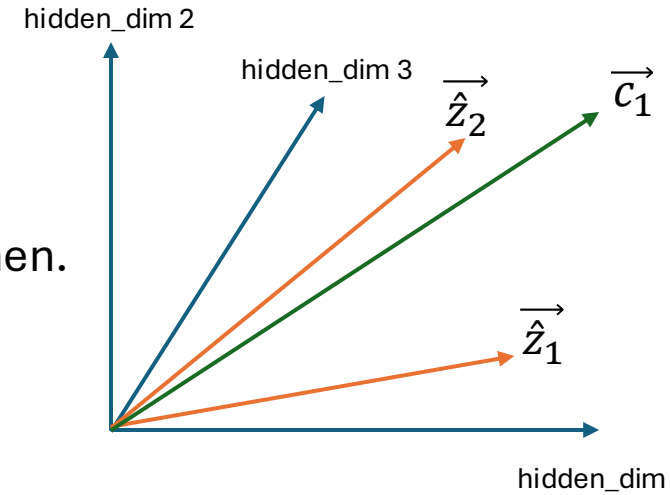
NB!!! Dimensjonene til Q,K og V har blitt transformert fra (batch_size,num_patches,hidden_dim) til (batch_size,num_heads,num_patches,head_dim) før multi-headed self-attention operasjonen

Steg 5:

$$C = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- C(batch_size,num_heads,num_patches,head_dim)=(32,2,1024,64)
  - num_heads: antall attention heads i Transformer-blocken = 2
  - head_dim: hidden_dim/num_heads = 128/2 = 64
- Q(batch_size,num_heads,num_patches,head_dim)=(32,2,1024,64)
- K^T(batch_size,num_heads,head_dim,num_patches)=(32,2,64,1024)
- QK^T(batch_size,num_heads,num_patches,num_patches)=(32,2,1024,1024): Attention score matrisa!
- d_k: head_dim = hidden_dim/num_heads = 128/2 = 64
- V(batch_size,num_heads,num_patches,head_dim)=(32,2,1024,64)

Her regner vi ut context vektorene ved bruk av multi-head attention mekanismen.



NB!!! Dette steget forklarte jeg feil i presentasjonen, vi regner ut logits direkte fra context vektoren gjennom en lineær transformasjon, vi bruker ikke embedding matrise

Steg 6:

$$\text{logits} = C A_{\text{down}}^{T} + b_{\text{down}}$$

- logits(batch_size,num_patches,num_tokens-1) = (32,1024,16)
  - Unormaliserte sannsynlighetsfordelinger for alle patches. Vi bruker num_tokens -1 i siste dimensjon, ettersom vi ikke skal gi sannsynlighet til masked patch.
- C(batch_size,num_patches,hidden_dim) = (32,1024,128)
  - Nå har vi transformert tilbake til (batch_size,num_patches,hidden_dim) fra (batch_size,num_heads,num_patches,head_dim)
- A_{down}(num_tokens-1,hidden_dim) = (16,128)
- b_{down}(num_tokens-1) = (16)

Logits

	[0,0,0,0]	[0,0,0,1]	[0,0,1,0]	[0,0,1,1]		[1,1,1,1]
$X_{1,1}$	1.4	-1.2	0.01	13	…	10.8
$X_{1,2}$	2.2	-0.1	3.2	6.8	…	-0.1
	⋮	⋮	⋮	⋮	…	⋮
$X_{32,32}$	4.2	-1.3	0.02	1.5	…	12.1

Steg 7:

$$P(X) = \text{Softmax}(\text{logits})$$

- $P(X)$ (batch_size,num_patches,num_tokens-1)
  - Sannsynlighetsfordeling for vær patch.

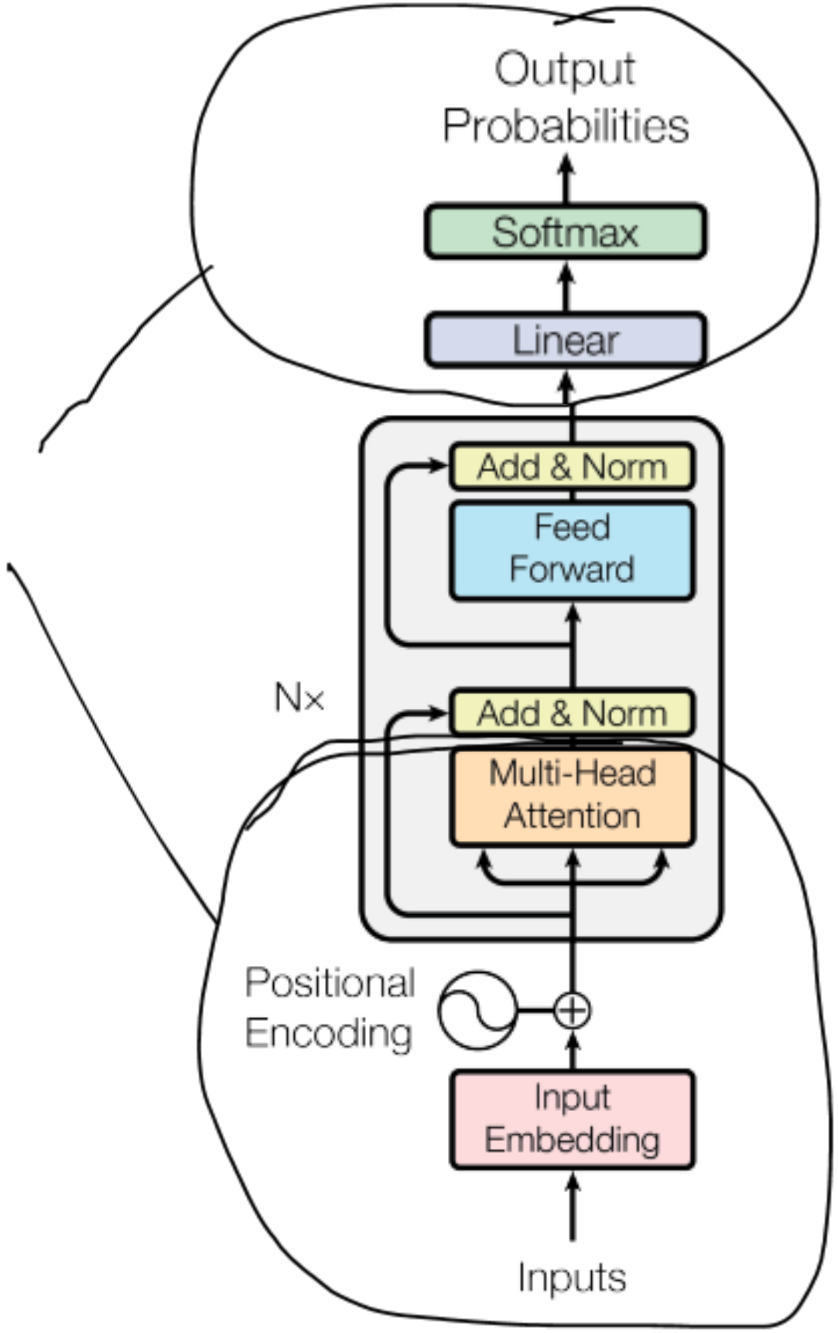
P(X)

	[0,0,0,0]	[0,0,0,1]	[0,0,1,0]	[0,0,1,1]		[1,1,1,1]
$X_{1,1}$	0.05	0.001	0.01	0.43	…	0.4
$X_{1,2}$	0.1	0.01	0.2	0.6	…	0.005
	⋮	⋮	⋮	⋮	…	⋮
$X_{32,32}$	0.05	0.0001	0.003	0.01	…	0.9

Oppsummering

- Det er noen ekstra steg i Transformer-blocken (normalization og FFN) som ikke er beskrevet for enkelhet.
- De finnes i det originale paperet: https://arxiv.org/pdf/1706.03762

Har beskrevet dette.




LOSS FUNCTION

Cross entropy loss:

$$L(P(X), Y) = -\frac{1}{N * B}\sum_{j=1}^{B}\sum_{i=1}^{N}\log(P(X)_{j,i}^{\text{unmasked}}[Y_{j,i}^{\text{unmasked}}])$$

- Y(num_patches*batch_size)=(1024*32)
  - Dette er de originale bildene i batchen (umaskert)
- $P(X)$(num_patches*batch_size,num_tokens-1)=(1024*32,16)
  - Sannsynlighetskart for det maskerte bildet (output av forward pass)
- $P(X)_{j,i}^{\text{unmasked}}$(num_tokens-1)=(16)
  - Sannslighetsfordeling for den i-te maskerte patchen av det j-te bildet i batchen. Her er det kun inkludert radene av $P(X)_j$ hvor vi har en maskert patch.
- $Y_{j,i}^{\text{unmasked}}$(1)
  - Dette er verdien i den i-te maskerte patchen for det j-te bildet i batchen.
- $P(X)_{j,i}^{\text{unmasked}}[Y_{j,i}^{\text{unmasked}}]$(1)
  - Dette er prediksjonssannyligheten for de faktiske verdien $Y_{j,i}^{\text{unmasked}}$

BACKPROPAGATION

Adam-optimizer

https://github.com/j-w-yun/optimizer-visualization

PARAMETERS

Parametere

$$\theta = (b_{\text{up}}, \text{A}_{\text{up}}, P, W_k, W_q, W_v, b_{\text{down}}, A_{\text{down}}, \theta_{FFN})$$

- Totalt antall parametere:
  - b_{up}: 128
  - A_{up}: 128*4 = 512
  - P: 1024*128 = 131,072
  - W_k: 128*128 = 16,384
  - W_q: 128*128 = 16,384
  - W_v: 128*128 = 16,384
  - b_{down}: 16
  - A_{down}: 128*16 = 2,048
  - θ_{FFN}: $b_1 + A_1 + b_2 + A_2$ = 512 + 512*128 + 128 + 512*128 = 131,712
- Totalt: 314,640 parametere