Vision Transformer (ViT)

Čo je to transformer?

- Transformer je neurónová sieť moderného typu
- Prvý transformer na spracovanie textu vynašli v roku 2018
- Prvý transformer na spracovanie obrazu vynašli v roku 2020 (vizuálny transformer – ViT)

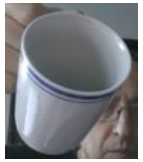
Čím je ViT výnimočný?

- Všetky predchádzajúce detektory objektov sa opierali o hrubú silu v tom zmysle, že najprv vytvorili klasifikátor (t.j. stroj, ktorý povie čo vidí) a potom ho pustili na veľa rôznych miest na obraze. Tam, kde klasifikátor niečo uvidel, tam bol objekt.
- **ViT to robí inak**

Klasický prístup

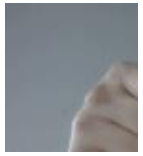
detektor

klasifikátor



šálka 0.98

šálka 0.21
okuliare 0.12
tvár 0.19



ruka 0.05

Ako funguje vizuálny transformer?

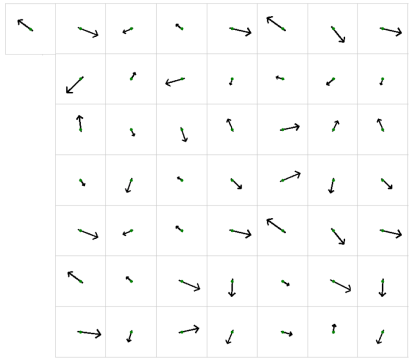
- VIT sa snaží premeniť obraz na príznakový vektor - rad čísel, z ktorých každé popisuje nejakú vlastnosť, ktorá rozlišuje jednotlivé obrazy navzájom.
- Tieto priznaky nemusia byť človeku pochopitelné, neurónová sieť si ich vytvára sama
- Pritom sieť trénujeme tak, aby významovo podobným obrazom dávala podobné príznakové vektory a rozdielnym rôzne

Ako funguje ViT?

Farebný obraz
3x224x224

32x7 = 224
7x7+1 = 50

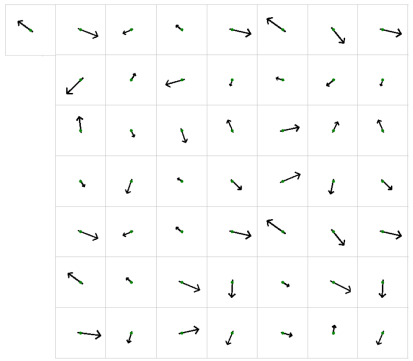
Skryté stavy
50x50x768



EMBEDDING
(vnorenie)

12x TRANSFORM
(transformácia)

Skryté stavy
50x50x768

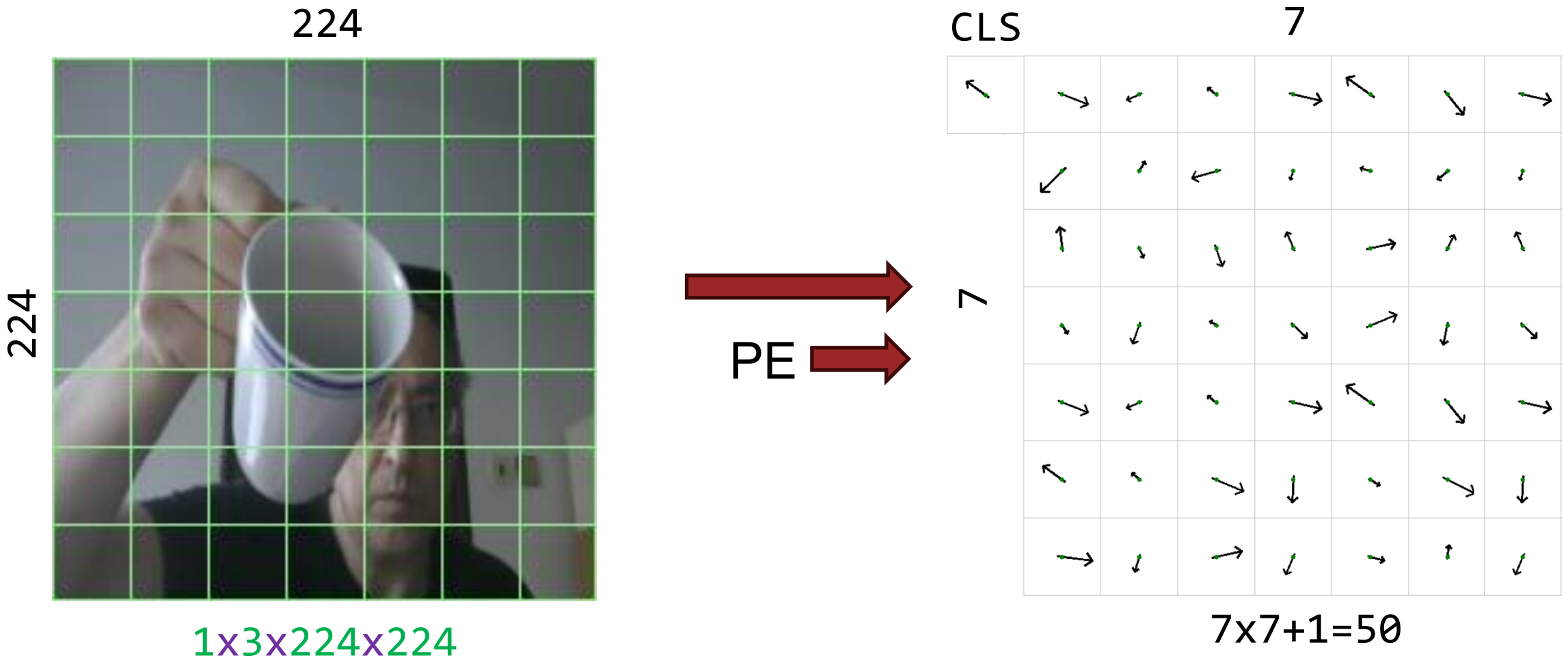


WIPEOUT
(vynorenie)

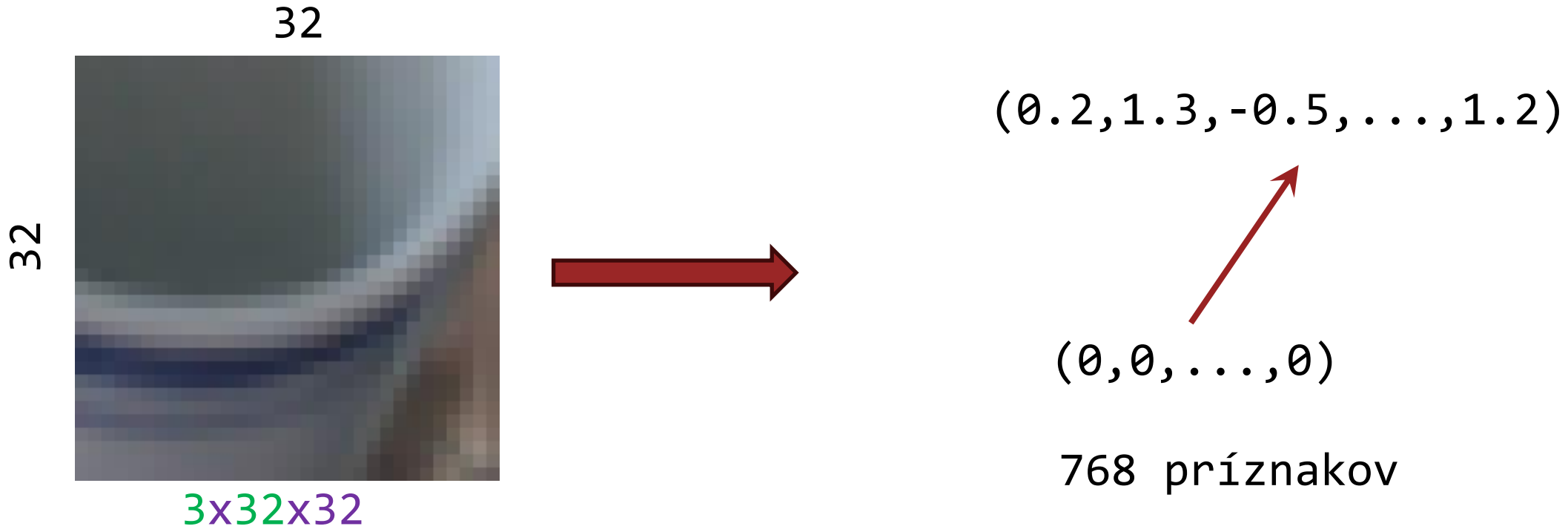
```
(0.1, 0.9, 0.25, …, 0.01)
```

ViT je klasifikátor, výstupom sú pravdepodobnosti príslušnosti obrázka k zvoleným kategóriám

- **Embedding**: obraz rozdelíme na políčka 32x32 a embeddingom 3072x768 (3x32x32=3072) premeníme na 7x7 vektorov dĺžky 768 (=12x64) a pridáme CLS

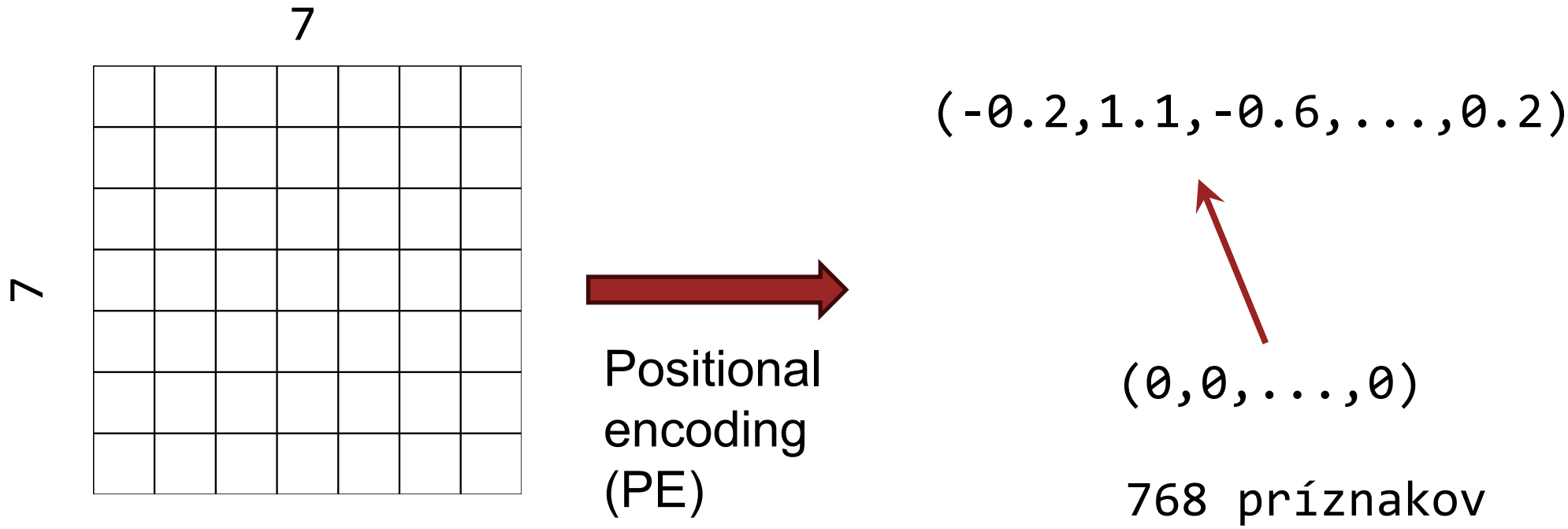


- Každé políčko s 32x32 pixelmi, z ktorých každý má tri farebné kanály (R,G,B) teda dohromady 3072 čísel sa premení na vektor dĺžky 768



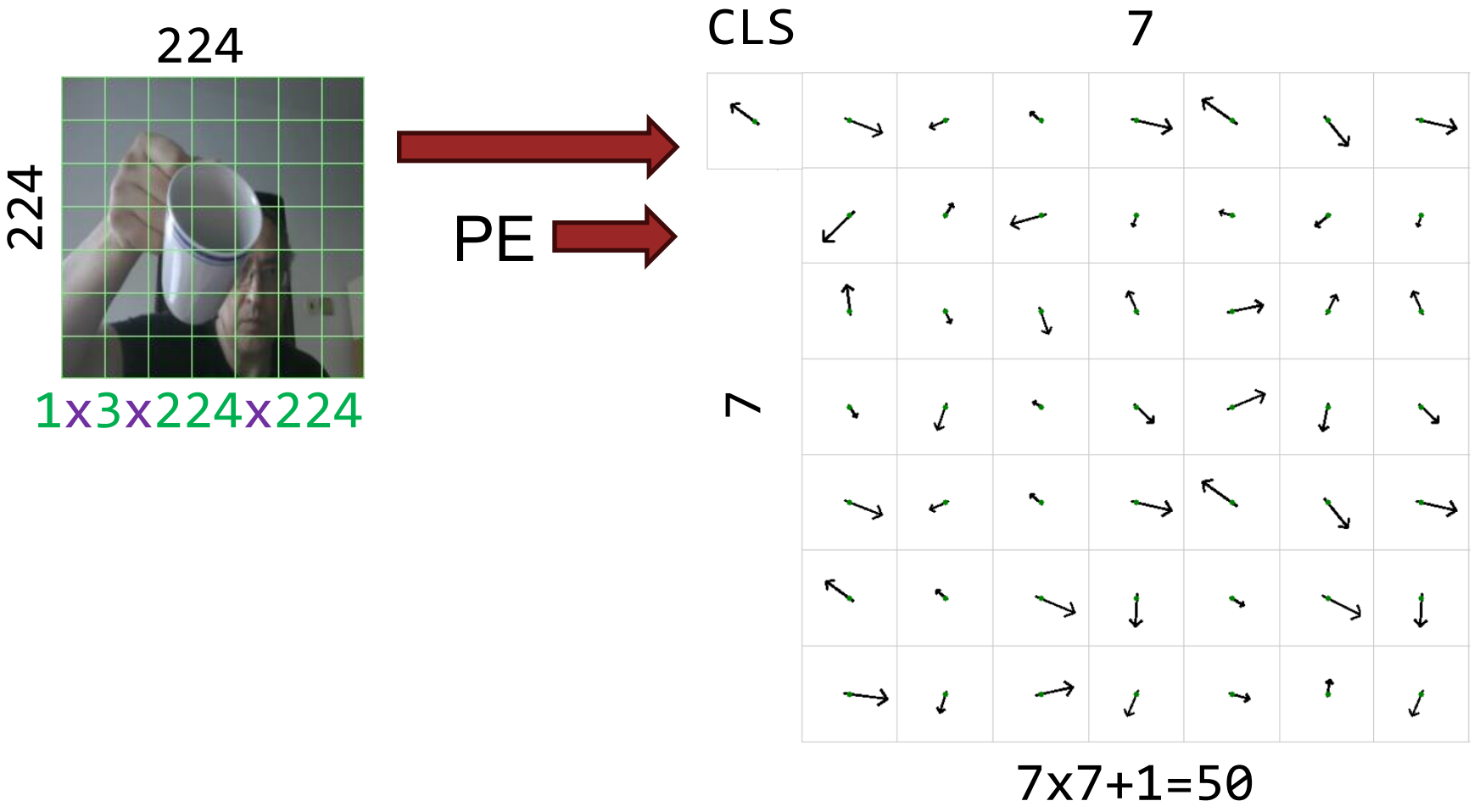
- Na túto premenu slúži spoločných 3072 x 768 konštánt, ktoré hovoria ako sa z jednotlivých hodnôt farieb políčka namieša každý jeden príznak vektora

- Ku každému políčku máme 768 čísel, ktoré kódujú jeho polohu; pritom pre políčka na podobnom mieste sú tieto čísla posobné



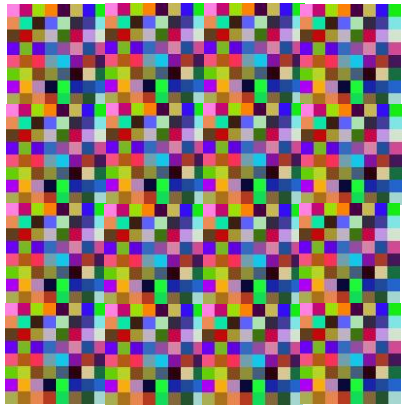
- Tieto pripočítavame k vektoru, ktorý reprezentuje políčko, takže výsledný vektor embeddingu je ovplyvnený obsahom i polohou políčka

- Takže embedding premení obraz na 7x7(=49) vektorov dĺžky 768
  K nim pridáme ešte jeden, tzv. klasifikačný (CLS)

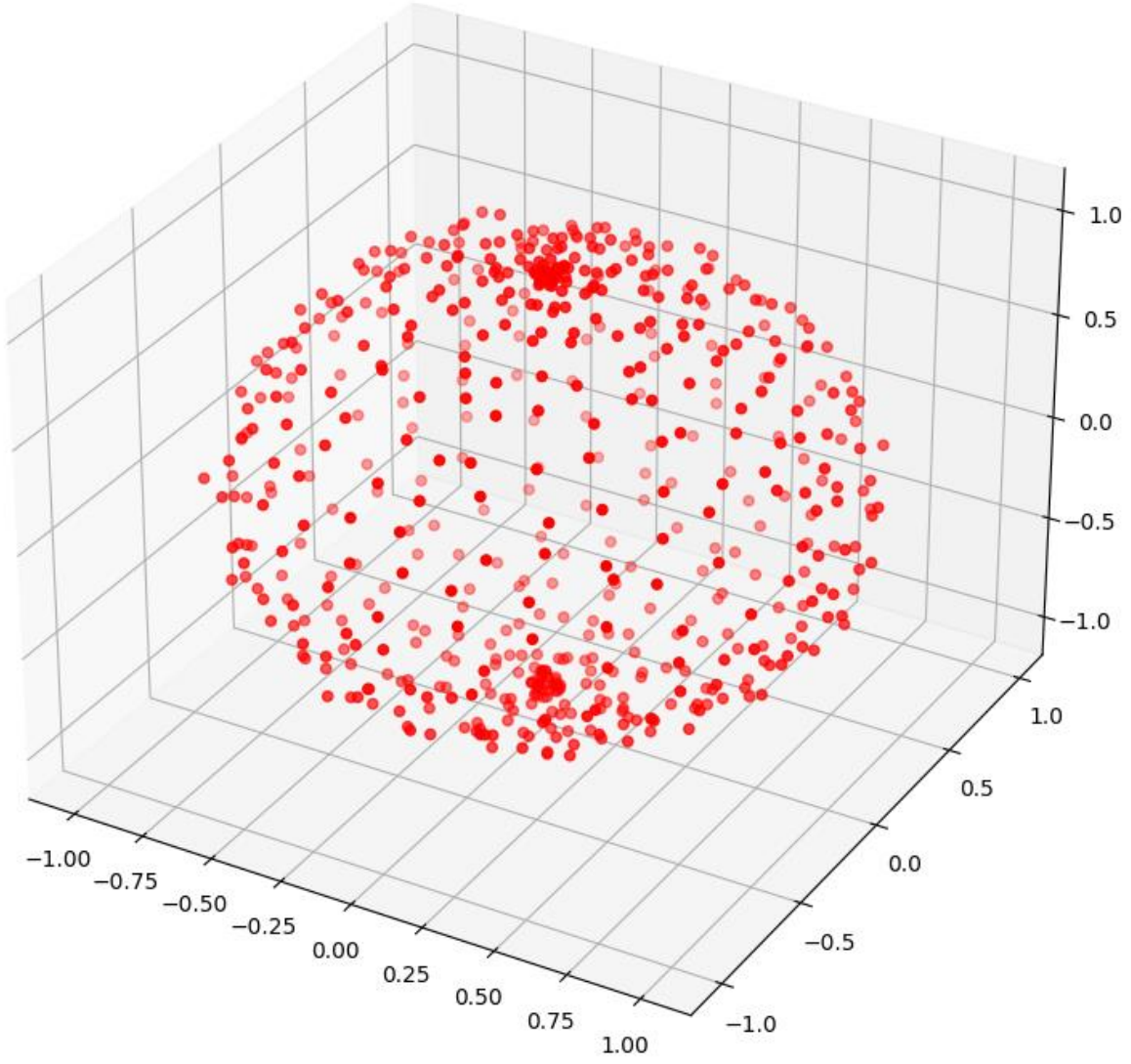


CLS

- Klasifikačný vektor je jediný skrytý stav, ktorý nemá konkrétne miesto na obraze. Jeho úlohou je vychytať v priebehu spracovania globálny význam



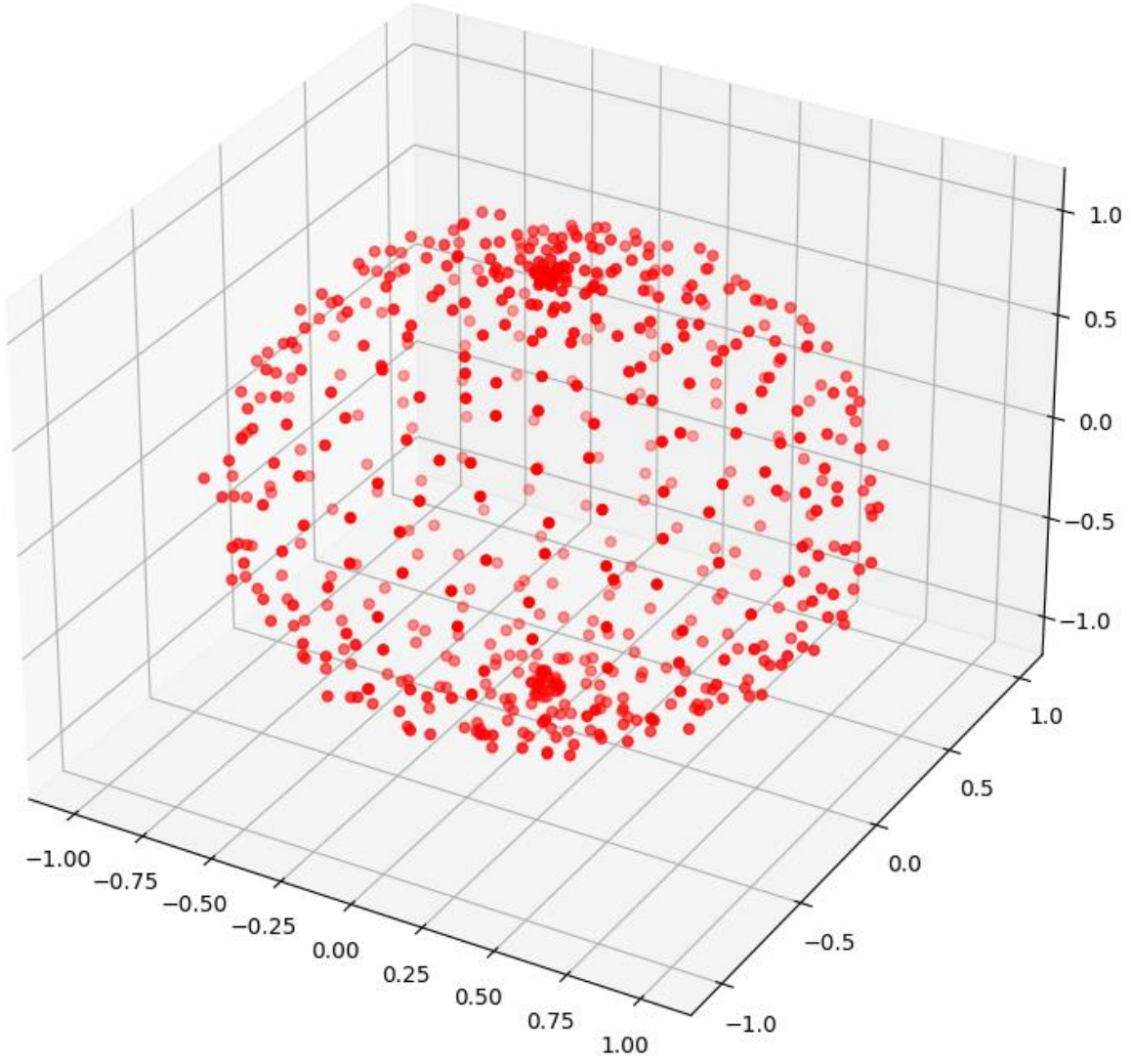
- výsledkom trénovania je, že počiatočná hodnota CLS nadobúda v priestore embeddingov konkrétnu hodnotu (reprezentujúcu „nevieme o čo ide“), ktorej dokonca môžeme spočítať inverziou embeddingu jej obrazovú podobu



- tieto vektory (nazývané skryté stavy) ukazujú zhruba k povrchu 768 rozmernej hypergule

- niektoré trčia von, iné dovnútra, pričom vzdialenosť od povrchu podlieha normálnemu rozdeleniu (distribúcia výchyliek zodpovedá zvonovej krivke)

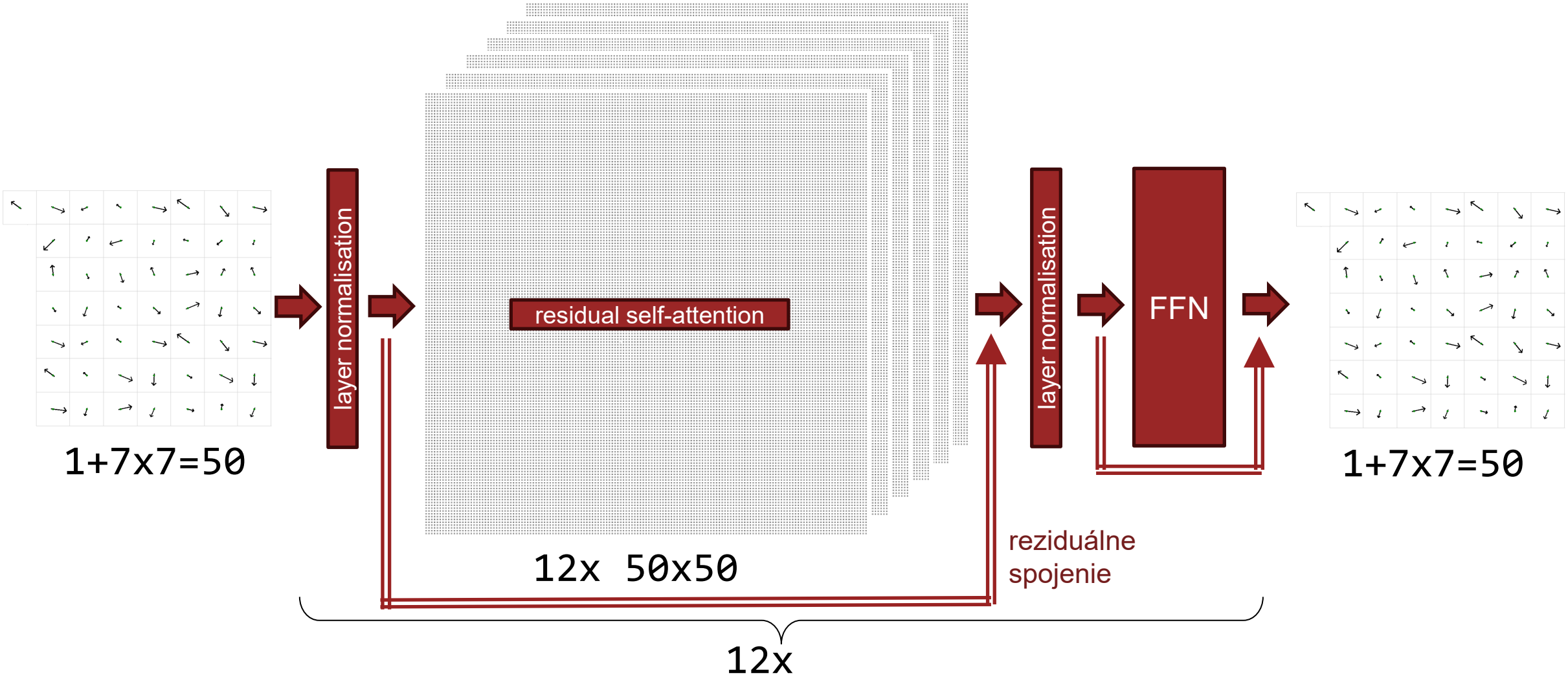
- v architektúre transformeru to je zabezpečené vkladaním stavebného prvku normalizácie vrstvy všade, kde sa dá



- Po natrénovaní siete, je uhol, ktorý vektory reprezentujúce políčka zvierajú, tým menší, čím častejšie sa spoločne vyskytujú v rovnakých obrázkoch datasetu, z ktorého trénujeme

- Významovo blízke políčka majú teda podobné vektory až na to, že to isté políčko môže mať rôzny význam v závislosti od toho, aké iné políčka sa s ním v obrázku nachádzajú

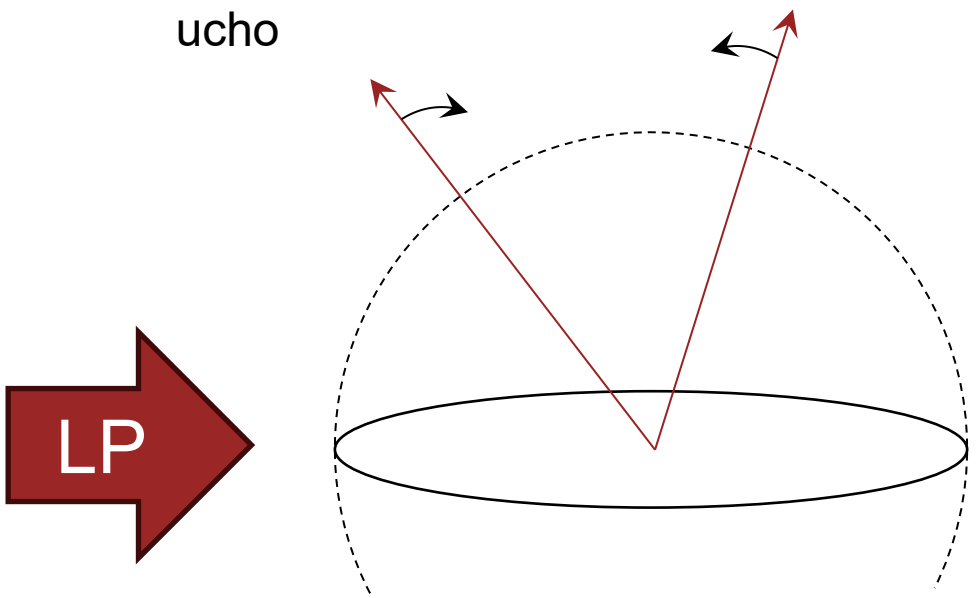
Transformačný blok



residual self-attention

Na políčku 10,10 vidíme špicaté ucho

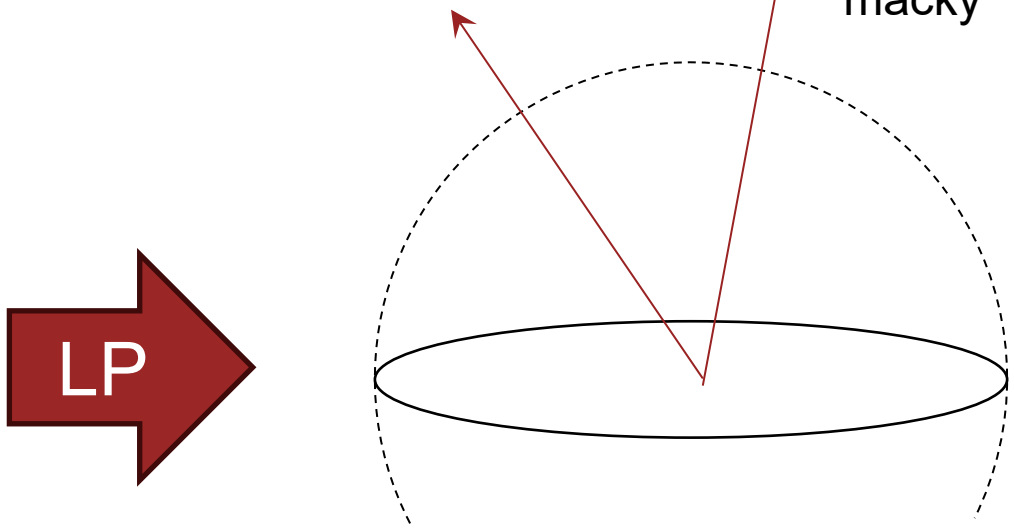
Na políčku 12,13 vidíme myš v drápoch



64 rozmerný priestor na jednej zo 6 hláv

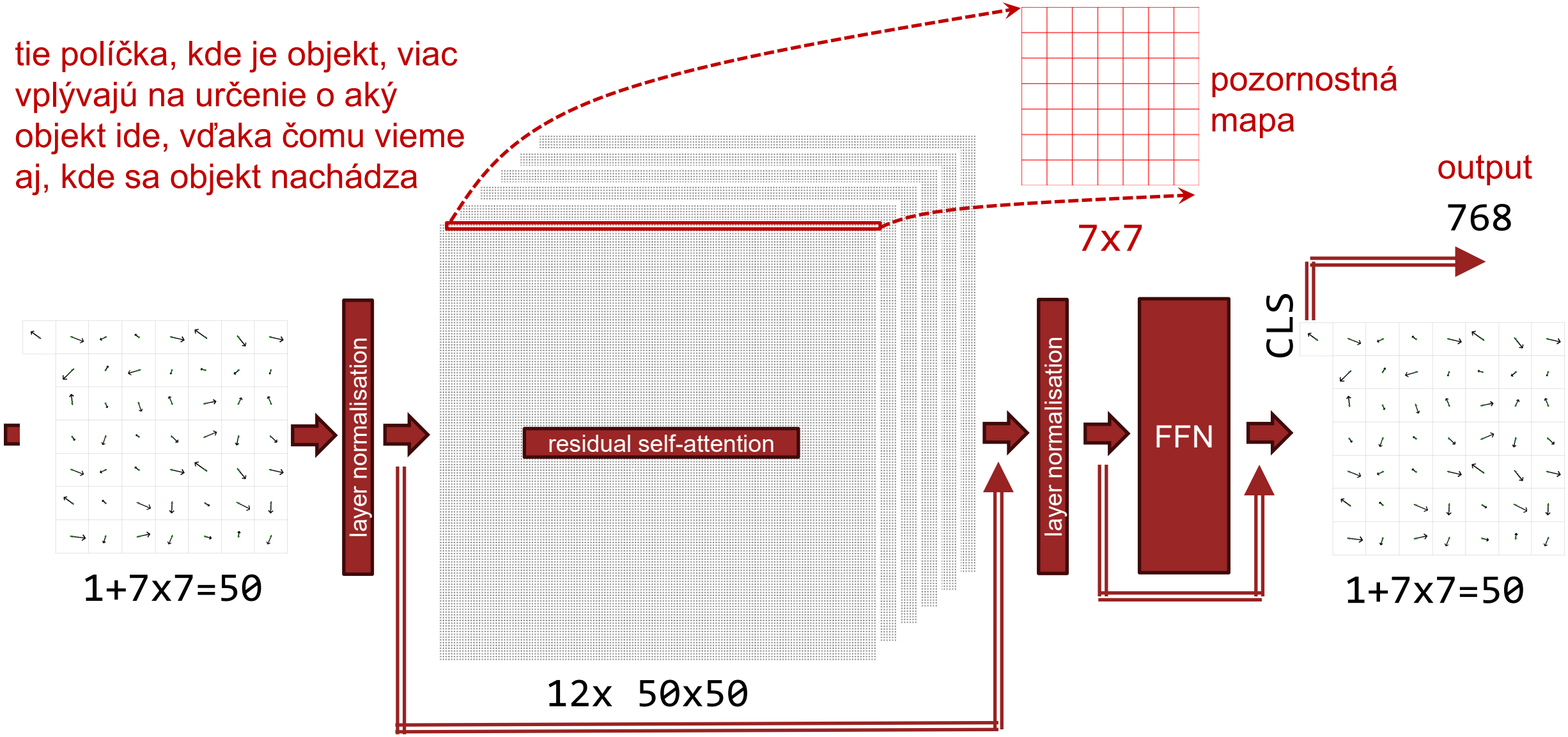
Na políčku 10,10 vidíme špicaté ucho mačky

Na políčku 12,13 vidíme ulovenú myš v drápoch mačky

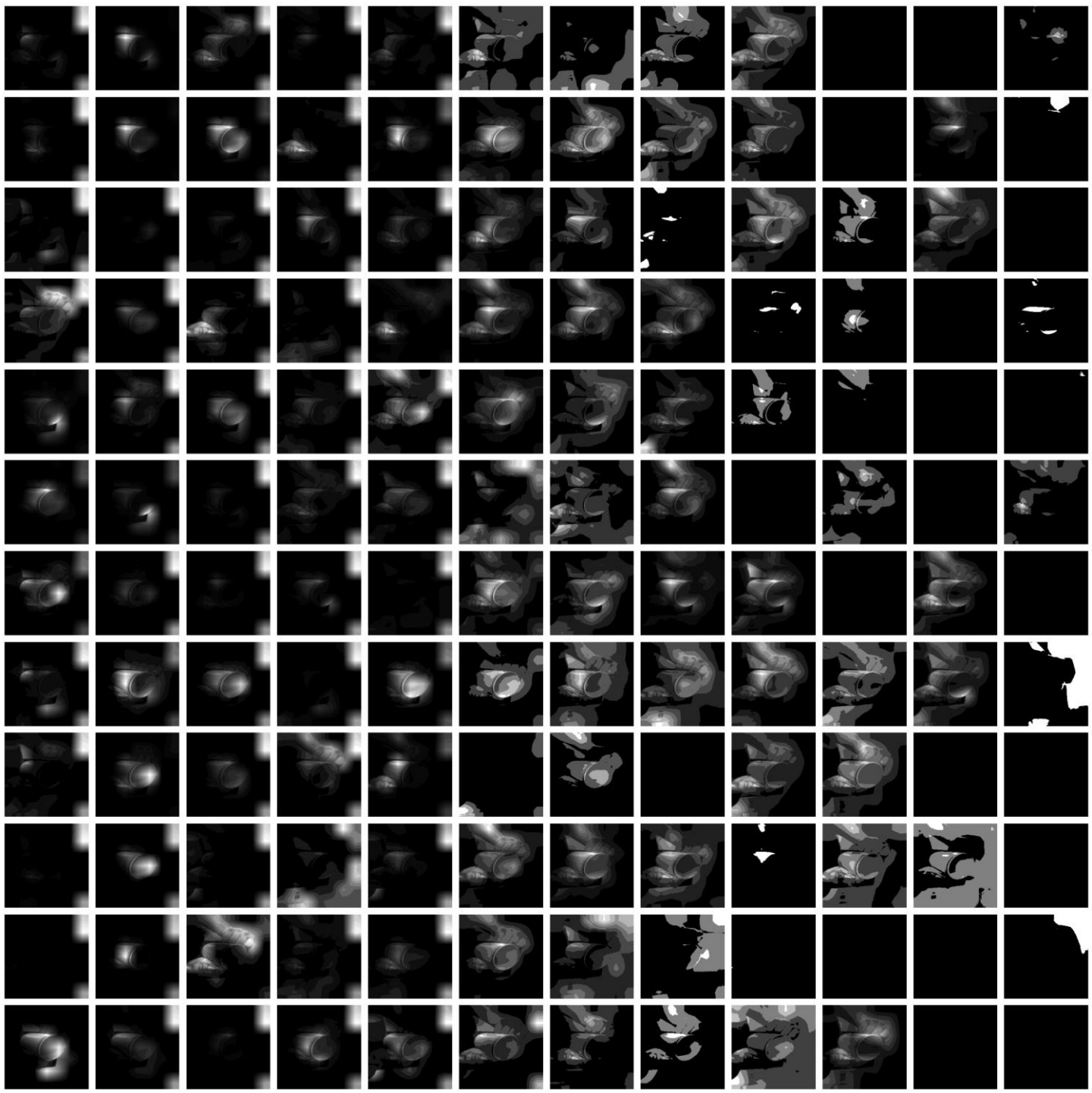


- *počítame ako sa má každý vektor posunúť v kontexte ostatných*
- *tým získavajú globálnejší význam*

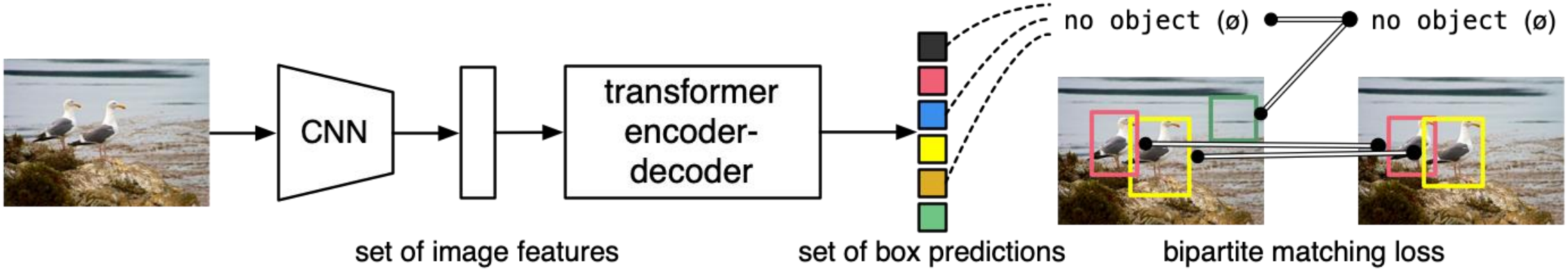
tie políčka, kde je objekt, viac vplývajú na určenie o aký objekt ide, vďaka čomu vieme aj, kde sa objekt nachádza



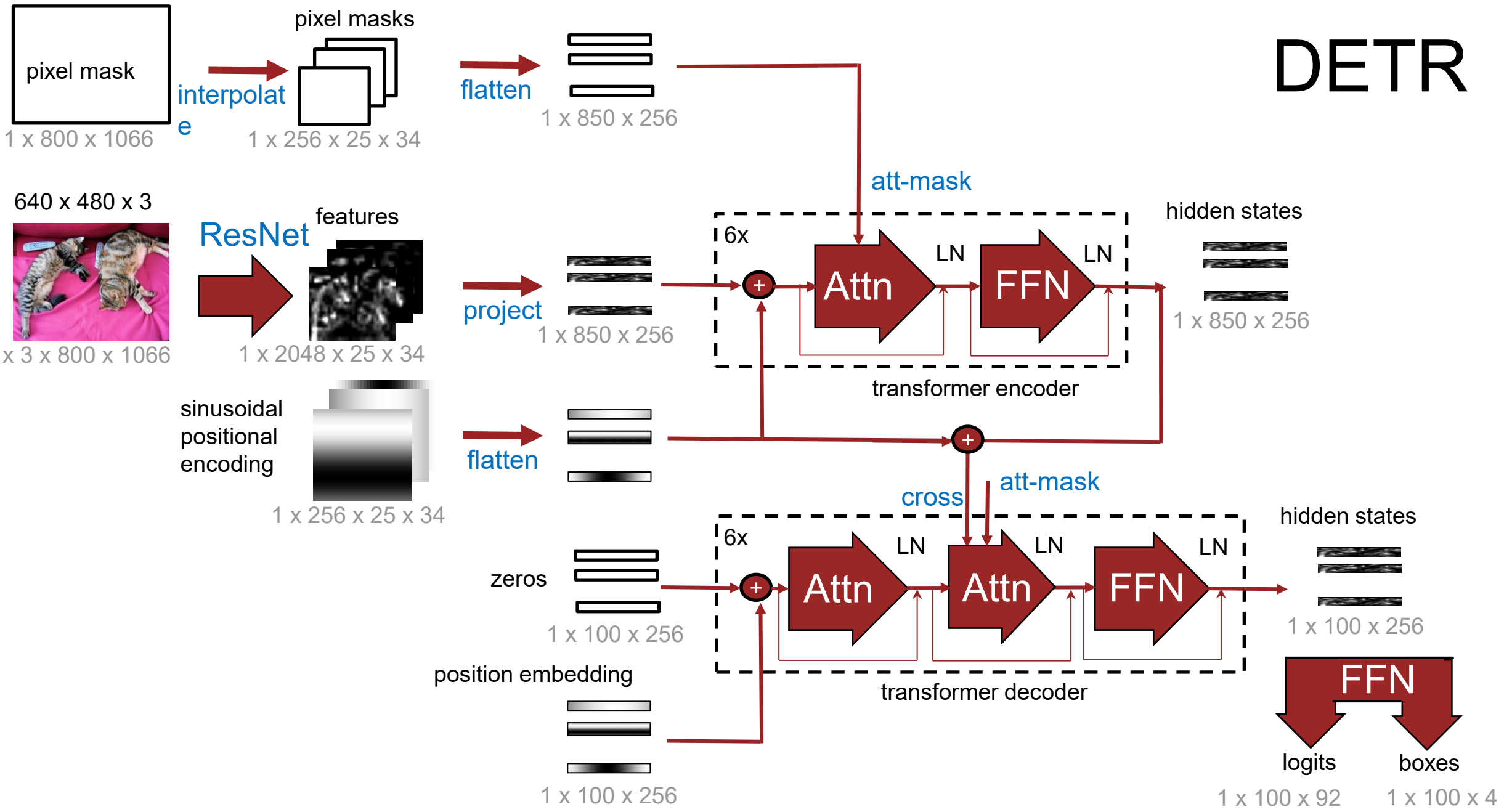
pozornostné mapy



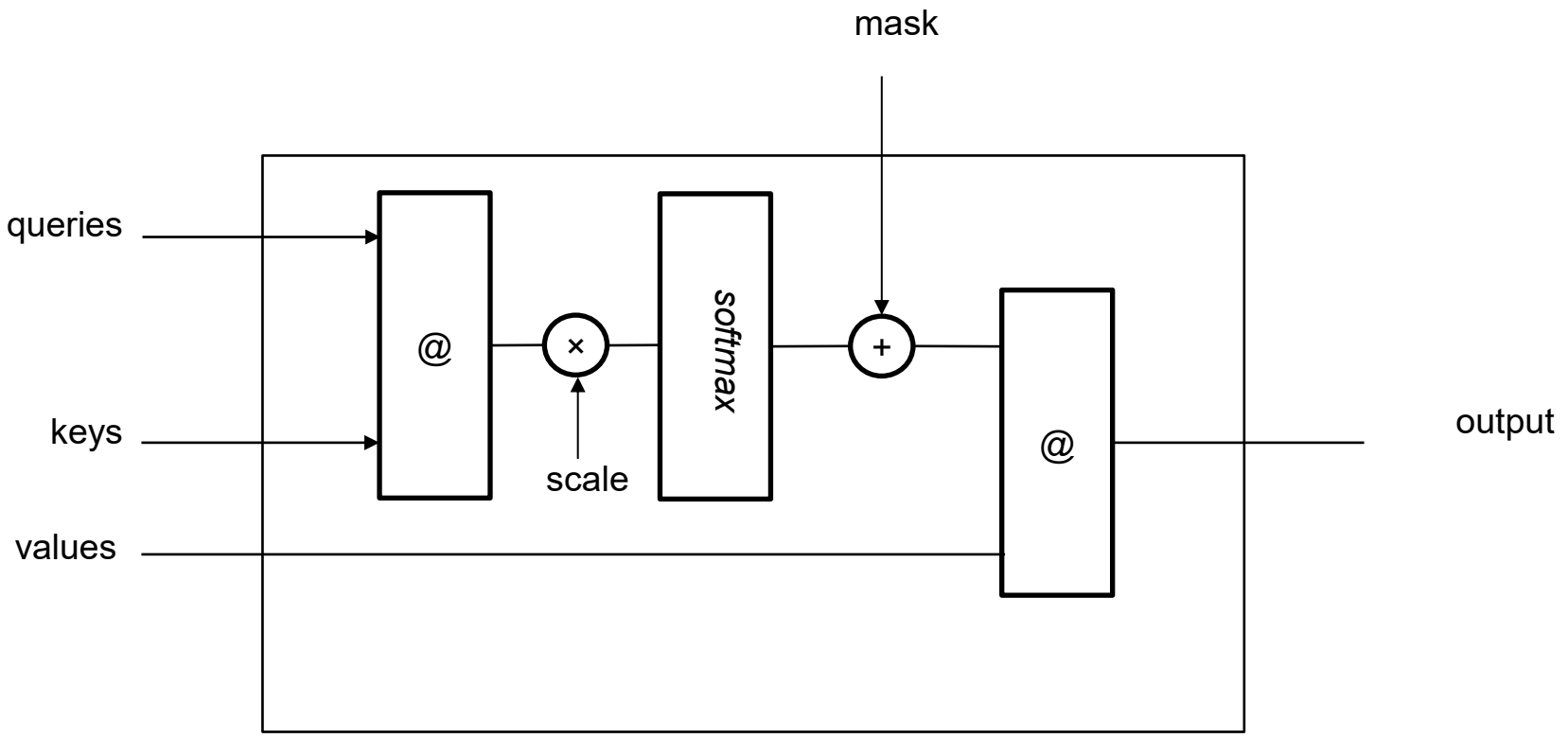
DETR (Detection Transformer)



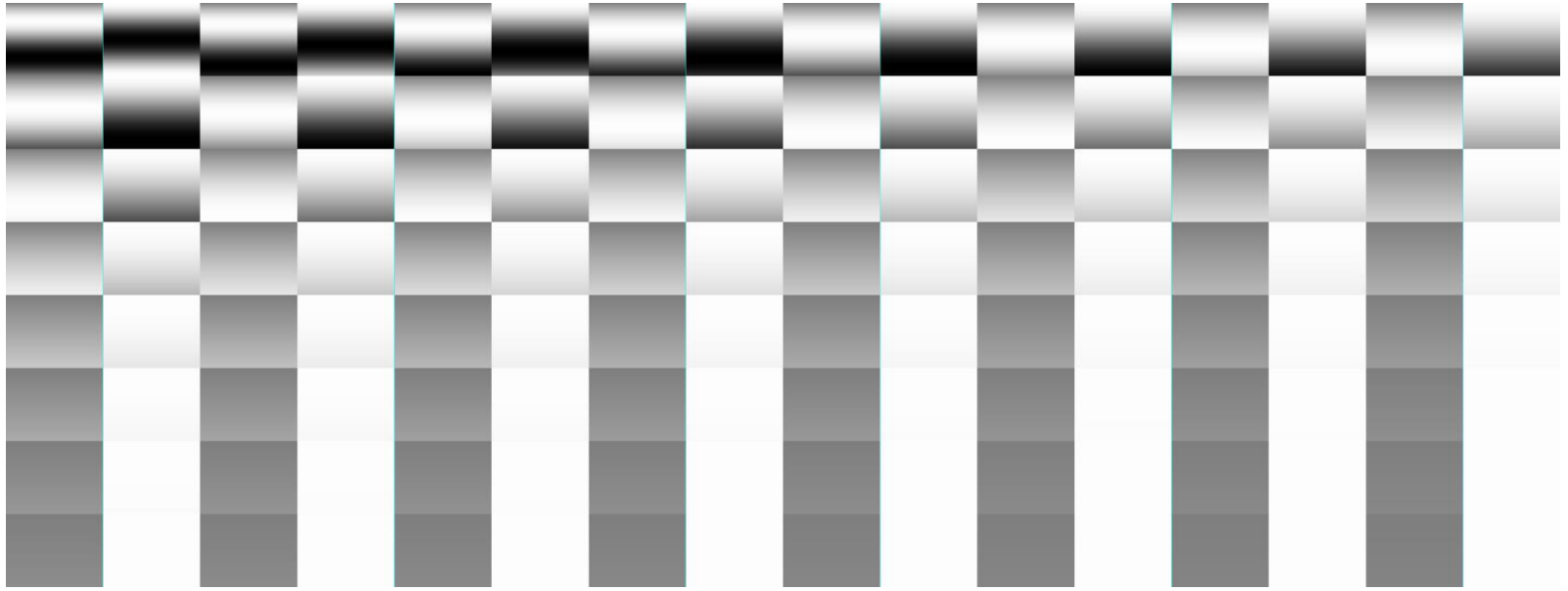
DETR

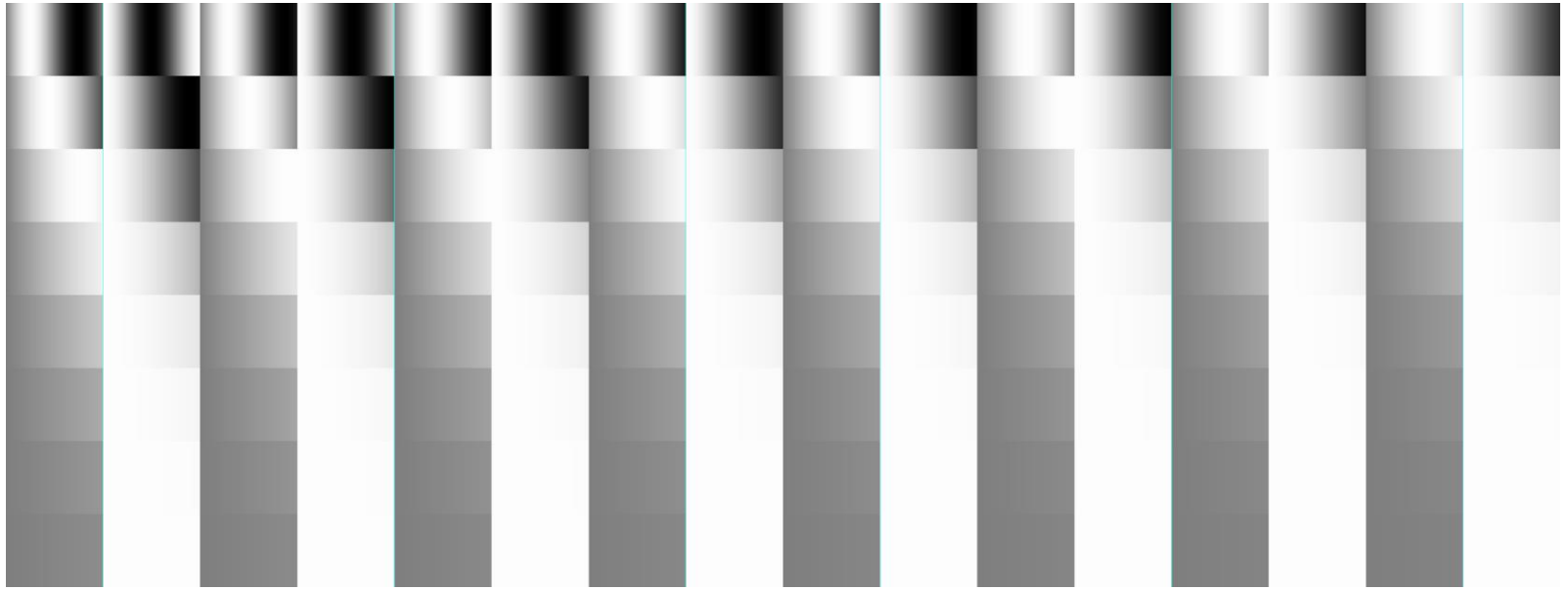

pixel mask
1 x 800 x 1066
interpolate
pixel masks
1 x 256 x 25 x 34
flatten
1 x 850 x 256
att-mask
640 x 480 x 3
1 x 3 x 800 x 1066
ResNet
features
1 x 2048 x 25 x 34
project
1 x 850 x 256
6x
Attn
LN
FFN
LN
hidden states
1 x 850 x 256
transformer encoder
sinusoidal positional encoding
1 x 256 x 25 x 34
flatten
cross
att-mask
zeros
1 x 100 x 256
position embedding
1 x 100 x 256
6x
Attn
LN
Attn
LN
FFN
LN
hidden states
1 x 100 x 256
transformer decoder
FFN
logits
1 x 100 x 92
boxes
1 x 100 x 4

Masked Attention



$$Att(Q, K, V) = \left[\text{softmax}\left(\frac{QK^T}{t\sqrt{d}}\right) + \text{mask}\right] V$$

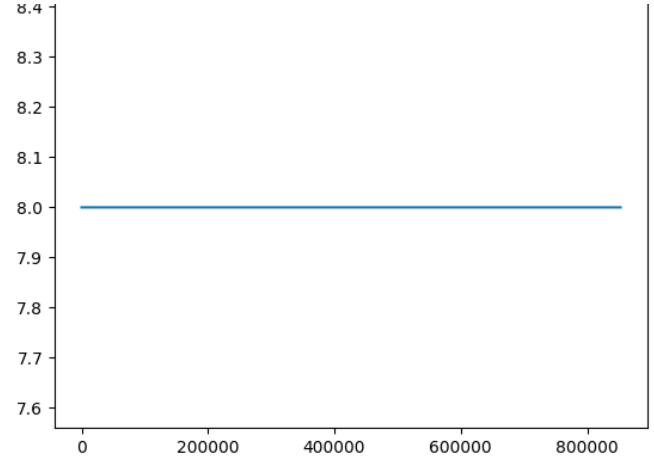




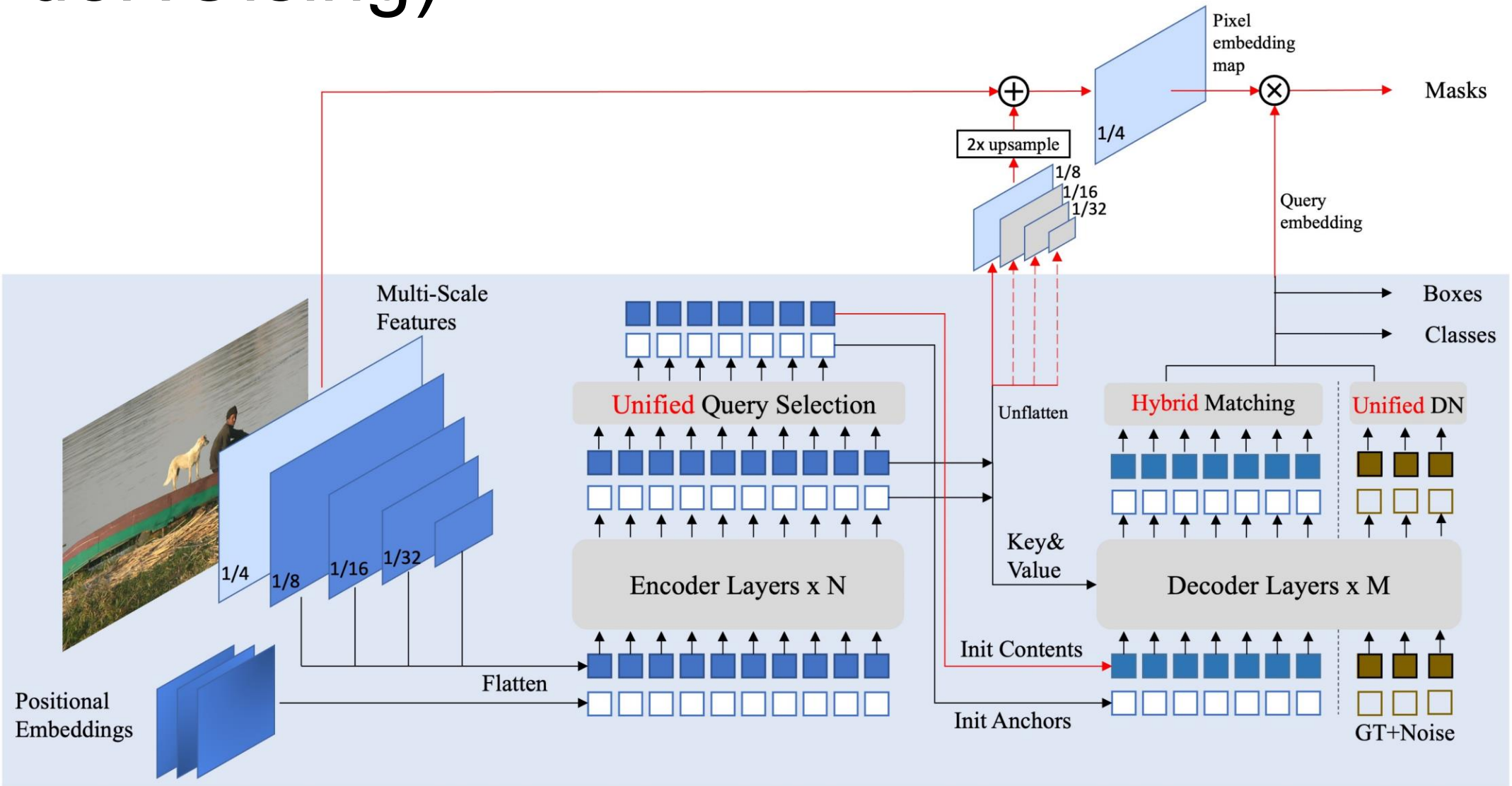
Sinusoidal positional encoding

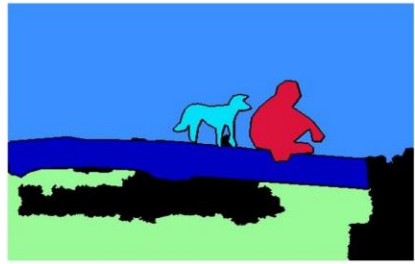
$$P(k, 2i) = \sin\left(\frac{k}{n^{2i/d}}\right)$$

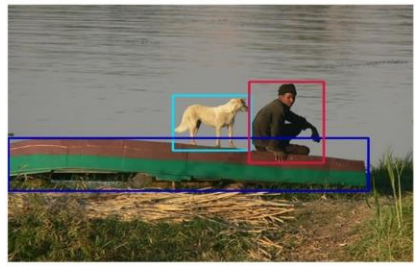
$$P(k, 2i+1) = \cos\left(\frac{k}{n^{2i/d}}\right)$$



Mask-DINO (Mask- DETR with Improved deNOising)



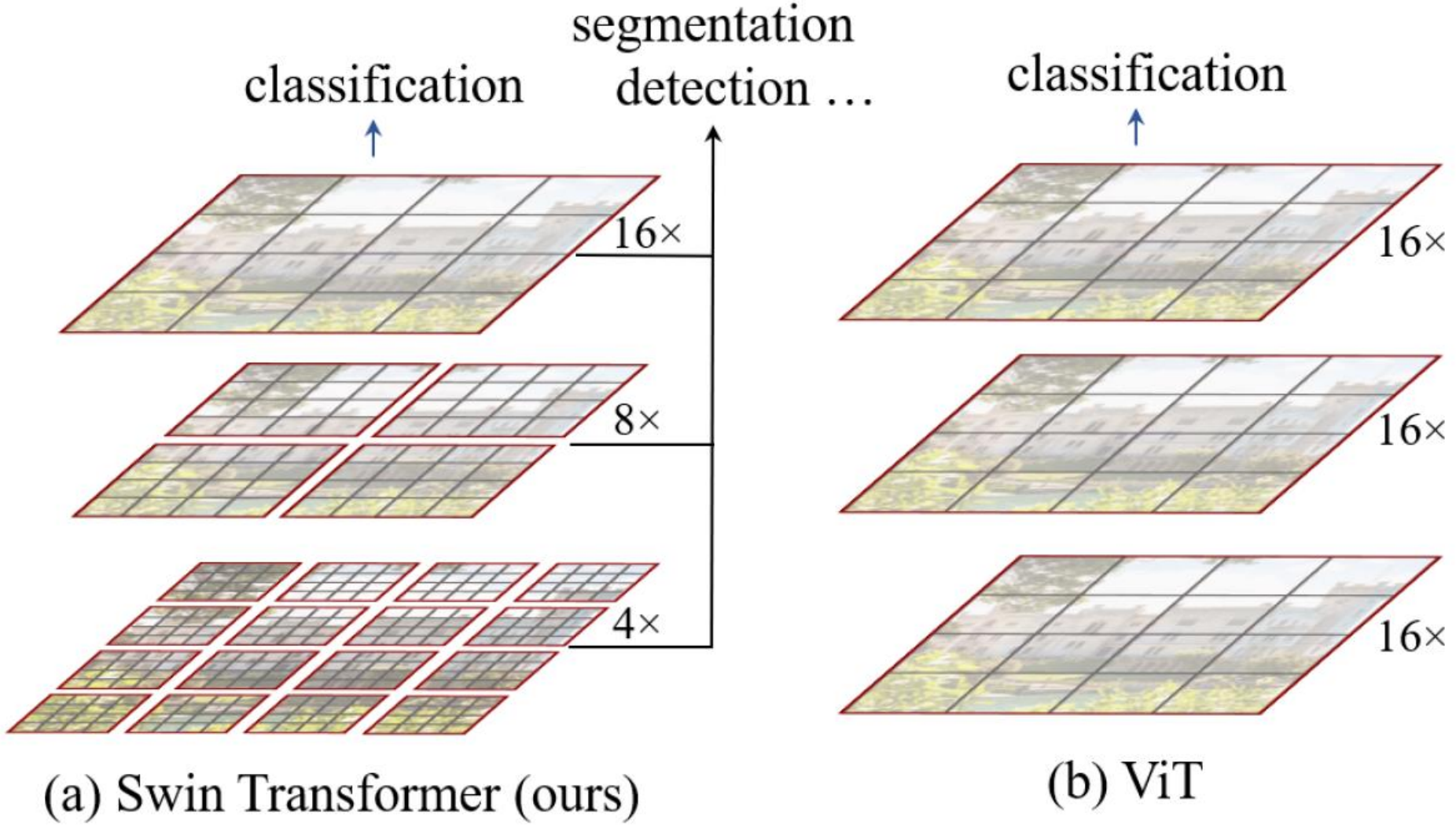




DN… Denoising

GT… Ground True

Swin (**S**hifted **Win**dows) Transformer



Swin Transformer

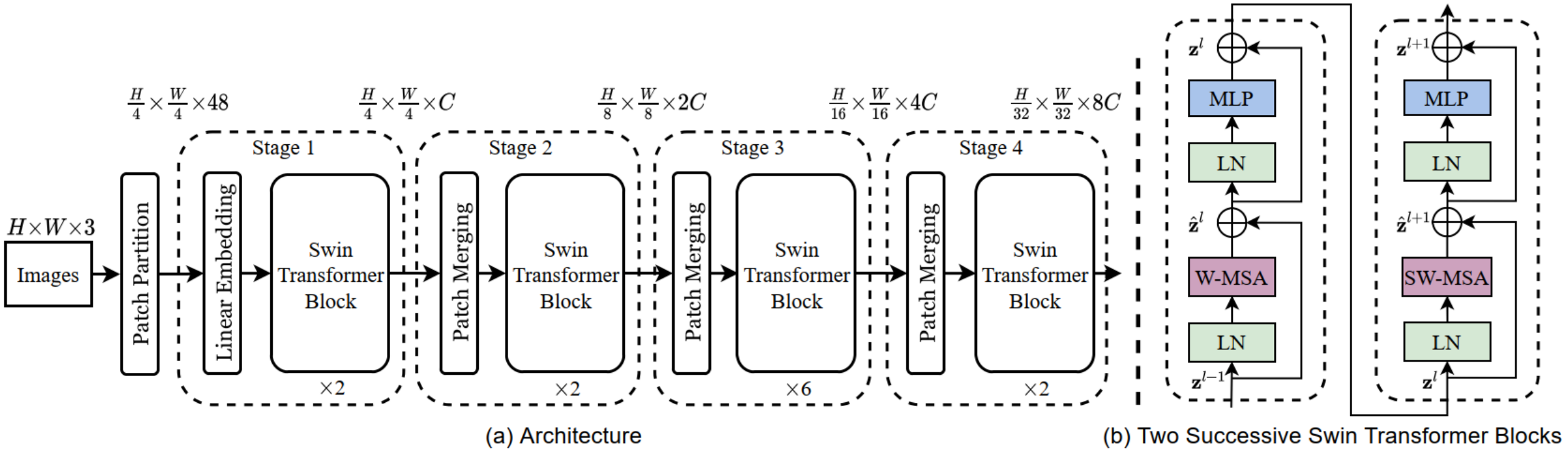


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.