Univerzitet u Nisu
Elektronski fakultet

Pregled i evaluacija vizuelnih transformera za klasifikaciju slika

DIPLOSMSKI RAD

MENTOR: PROF. DR ALEKSANDAR MILOSAVLJEVIĆ

STUDENT: ALEKSA MILIĆ, 17774

Sadrzaj

Racunarski vid i klasikacija slika

Racunarski vid

Sa inženjerske tačke gledišta, cilj je automatizovati zadatke koje ljudski vizuelni sistem može obavljati. Zadaci računarskog vida uključuju metode za pribavljanje, obradu i razumevanje digitalnih slika.

(https://en.wikipedia.org/wiki/Computer_vision)

Klasifikacija Slika

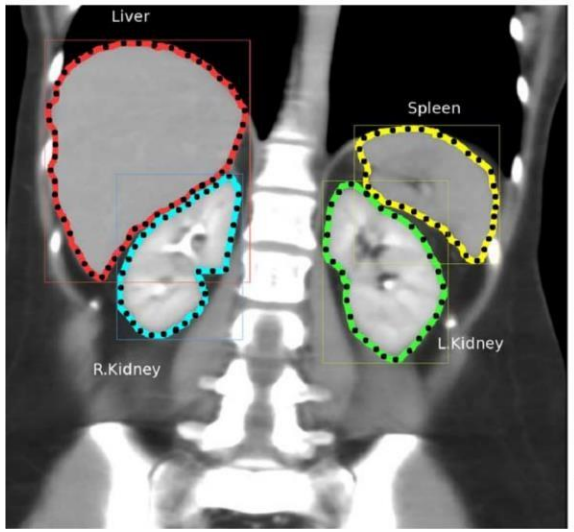
Klasifikacija slika je metod razvrstavanja slika na osnovu informacija iz njihovog konteksta. To u osnovi znači analiziranje i tumačenje veza između susednih piksela na slici. Glavni cilj ove metode je da se slike kategorišu koristeći ove kontekstualne informacije.

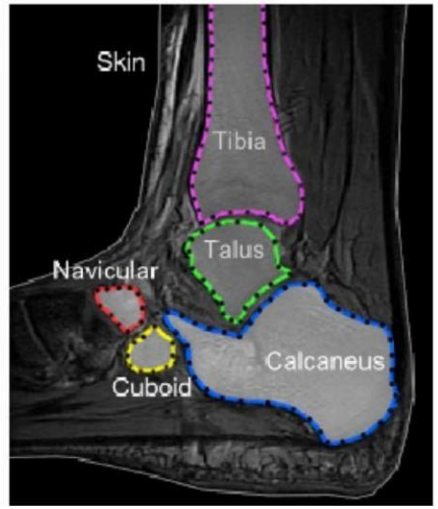
(https://en.wikipedia.org/wiki/Contextual_image_classification)

Sadrzaj

Racunarski vid: cemu?

- Skoro 80% podataka koji putuju internetom su vizualni podaci
- Danas svako ima telefon, i svaki telefon ima **bar 2 kamere**
- "**Slika vredi hiljadu reci**" je poznata izreka
- Ima **siroka** polja primene:
  - ✓ Robotika
  - ✓ Nadzor
  - ✓ Razne industrije
  - ✓ Samovozeći automobili/dronovi/autobusi
  - ✓ Medicina





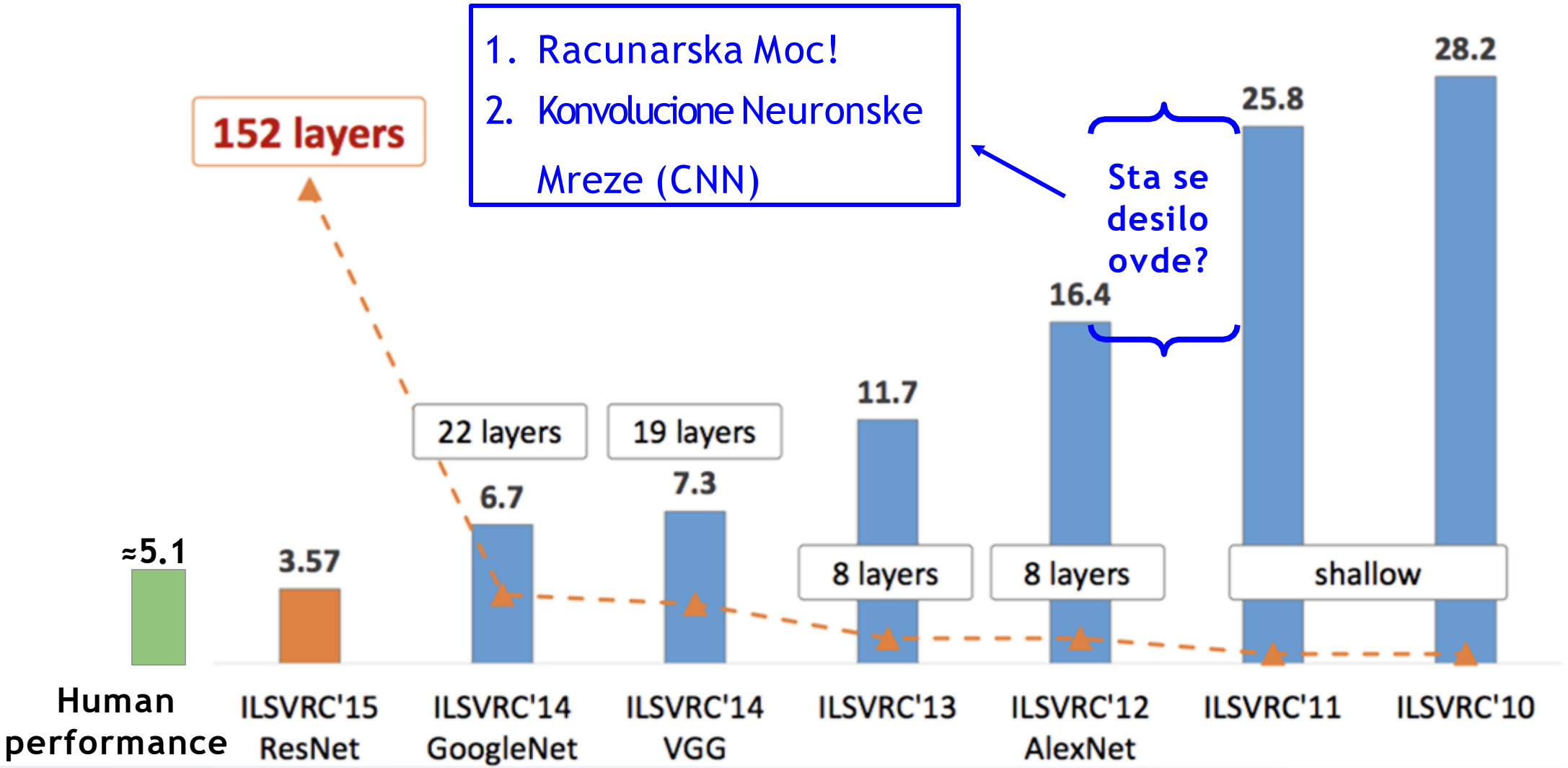
http://crcv.ucf.edu/people/faculty/Bagci/research.php

Racunarski vid: "hype"?

- Uzmimo u obzir **image-net**
- Samo jedan skupodataka
- (*http://www.image-net.org/challenges/LSVRC/*)
- 1000 klasa
- Vise od 1 milion slika

- Iako se ne zna tacan broj slika na internetu, procenjuje se da se broj meri u stotinama milijardi

Racunarksi vid:"hype"?



Racunarski vid - zadaci:

- **1. <u>Klasifikacija</u>**
- ***Sta*** je na slici?
  - Ljudi
  - Auto
  - Svetla semafora
  - Sat
  - …

http://vision.stanford.edu

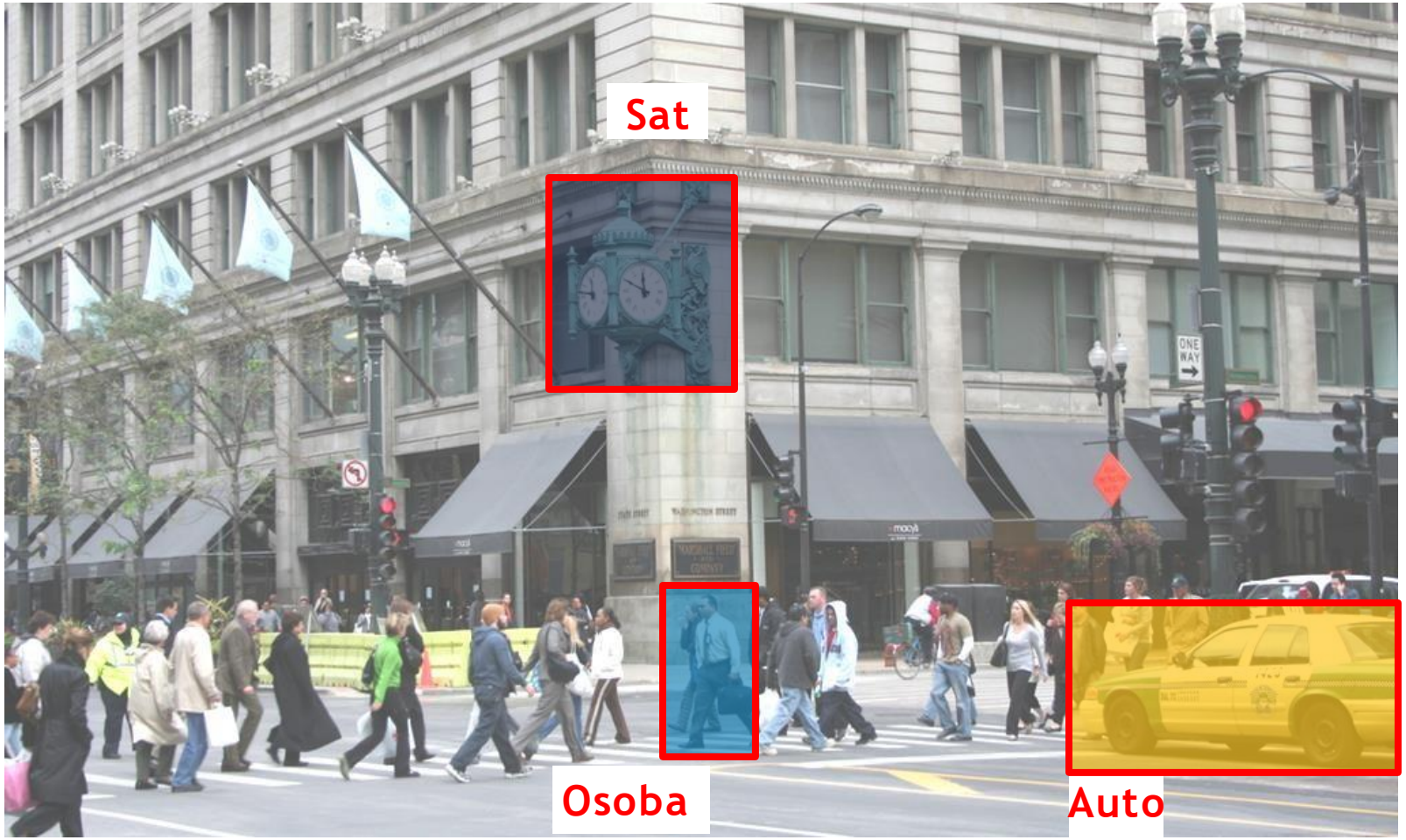
Racunarski vid-zadaci:

2. Detekcija

Sta je na fotografiji?

Gde se nalazi?

- Automobil u zutom pravouganiku
- Osoba u plavom
- Sat u sivom



http://vision.stanford.edu

Racunarski vid-zadaci:

3. Segmentacija

Sta' je na fotografiji?

Gde se nalazi?

Koje sve piksele sadrzi?

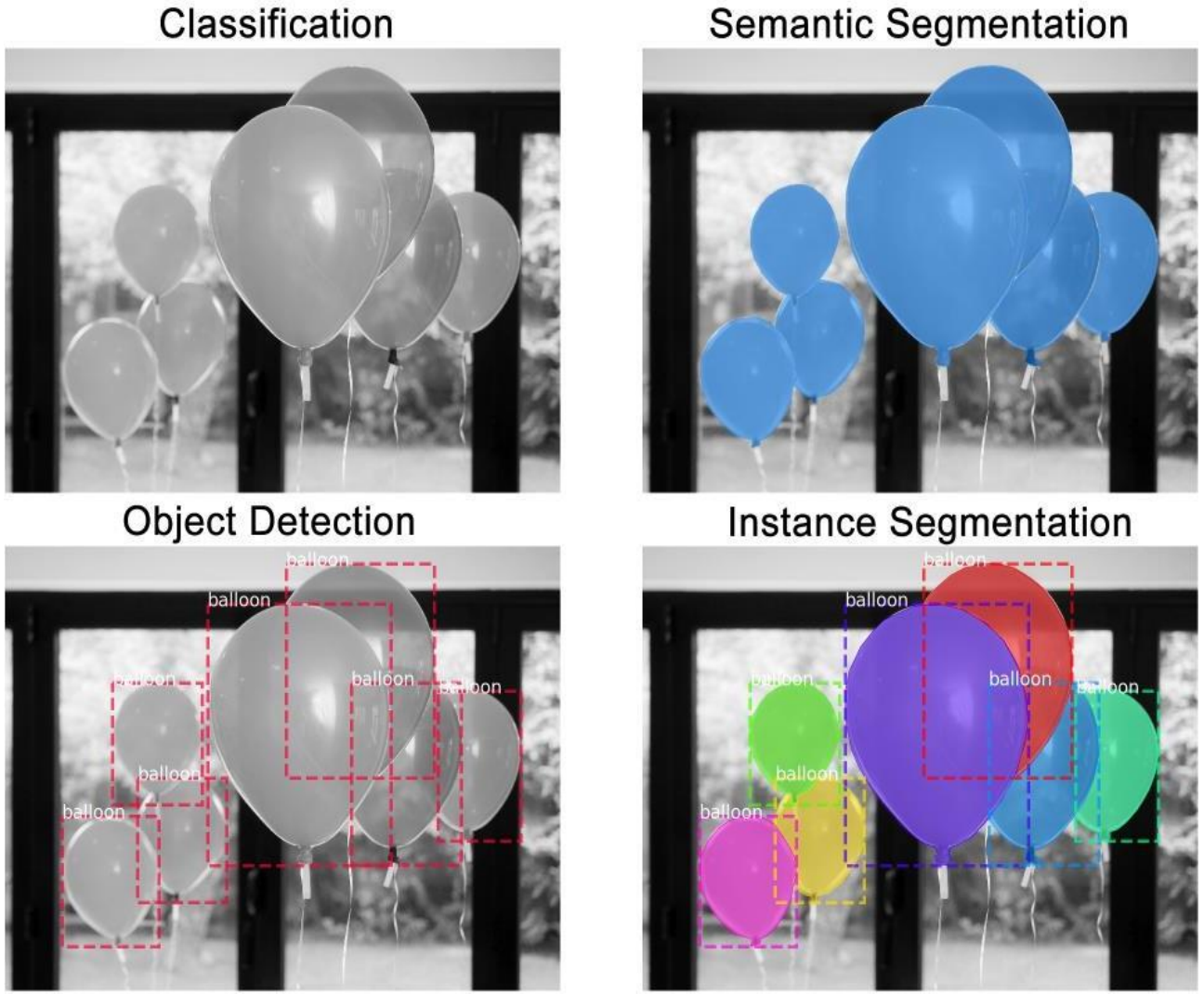
http://vision.stanford.edu

Racunarski vid-zadaci:

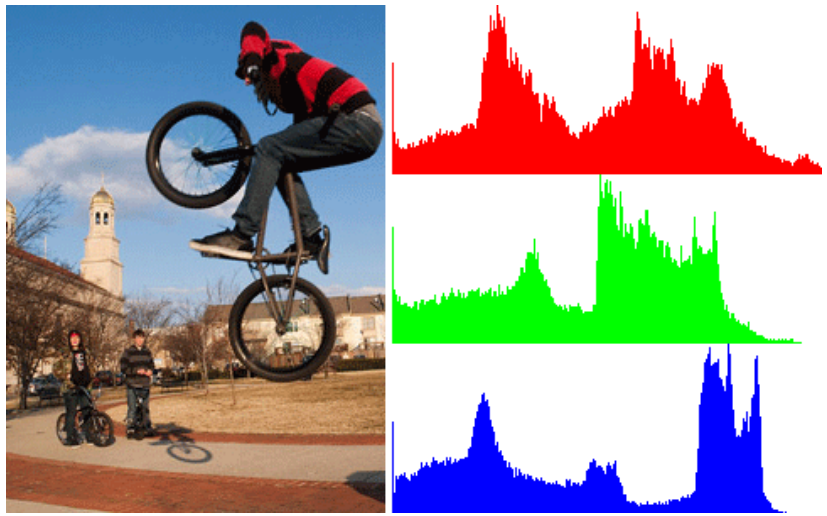
https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46

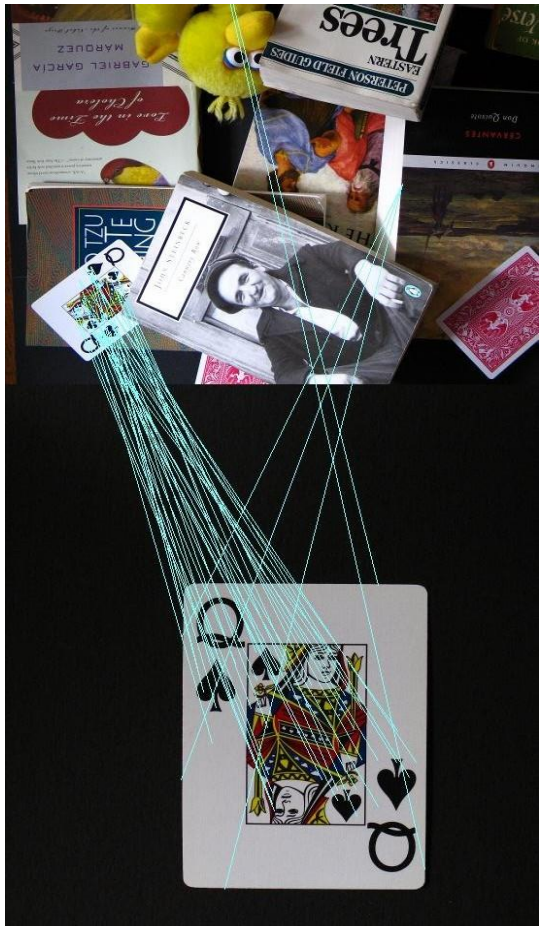
5. …i mnogi drugi

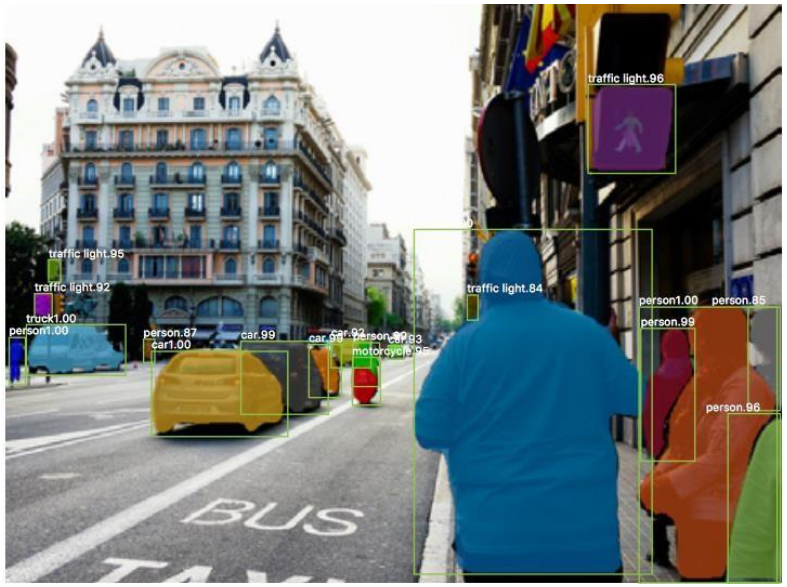
- Detekcija lica
- Detekcija osmeha
- Detekcija pogleda ociju



Sledeci koraci?







Digitalna obrada Osnovne operacije

Klasican pristup Inženjering karakteristika

Duboko ucenje Konvolucione neuronske mreze, Transformeri

Racunarski vid-zadaci:

4. Anotacija

Kako bi ste opisali datu sliku?

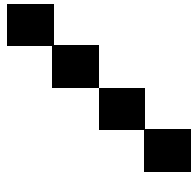
"Ljudi prelaze ulicu
dok automobil čeka"

http://vision.stanford.edu

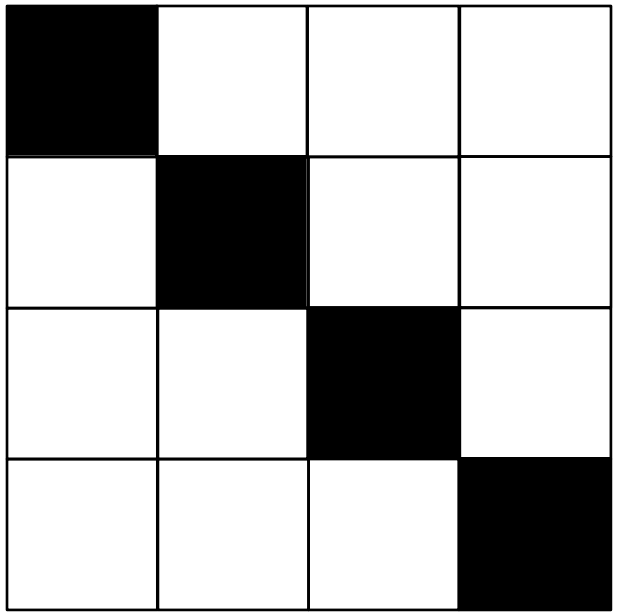
Sadrzaj

Reprezentacija slika

Slika, unutar racunara, je samo **matrica brojeva**



- 1: bela
- 0: cma
- Dimenzije: 4x4x1 matrica
- Dubina boje: 1 bit
- Kanali boje: 1
- Broj piksela: 16

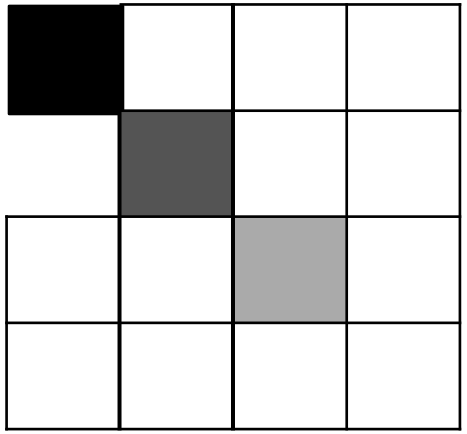


0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0

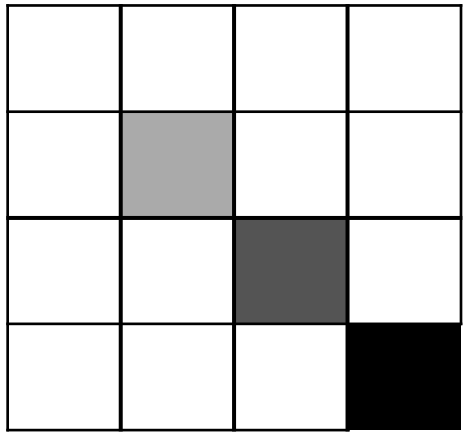
Reprezentacija slika

Slika, unutar racunara, je samo **matrica brojeva**

- 255: bela
- 0: crna
- Dimenzionalnost :**4x4x2** matrica
- Dubina boje: 8 bit
- Kanali boja: 2
- Broj piksela: **32**



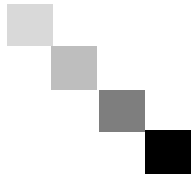
255	255	255	255
255	170	255	255
255	255	85	255
255	255	255	0



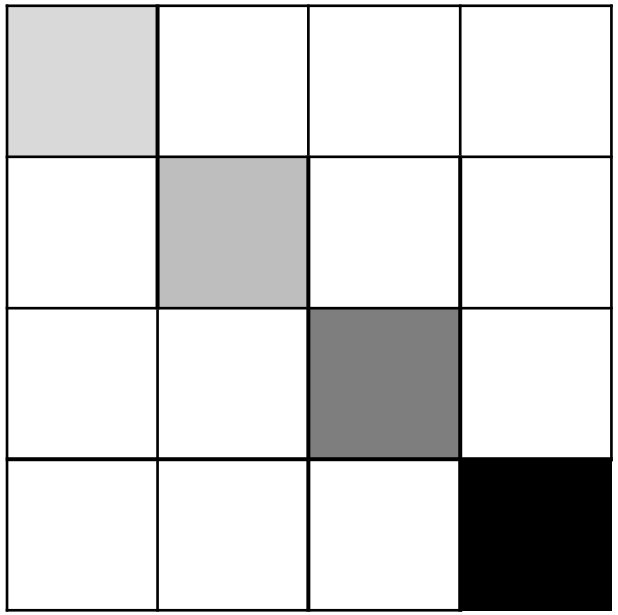
0	255	255	255
255	85	255	255
255	255	170	255
255	255	255	255

Reprezentacija slika

Slika, unutar racunara, je samo **matrica brojeva**



- **255**: Bela
- 0: Cma
- Dimenzionalnost:4x4x1 matrica
- Dubina boje: **8 bit**
- Kanali boja: 1
- Broj piksela: 16

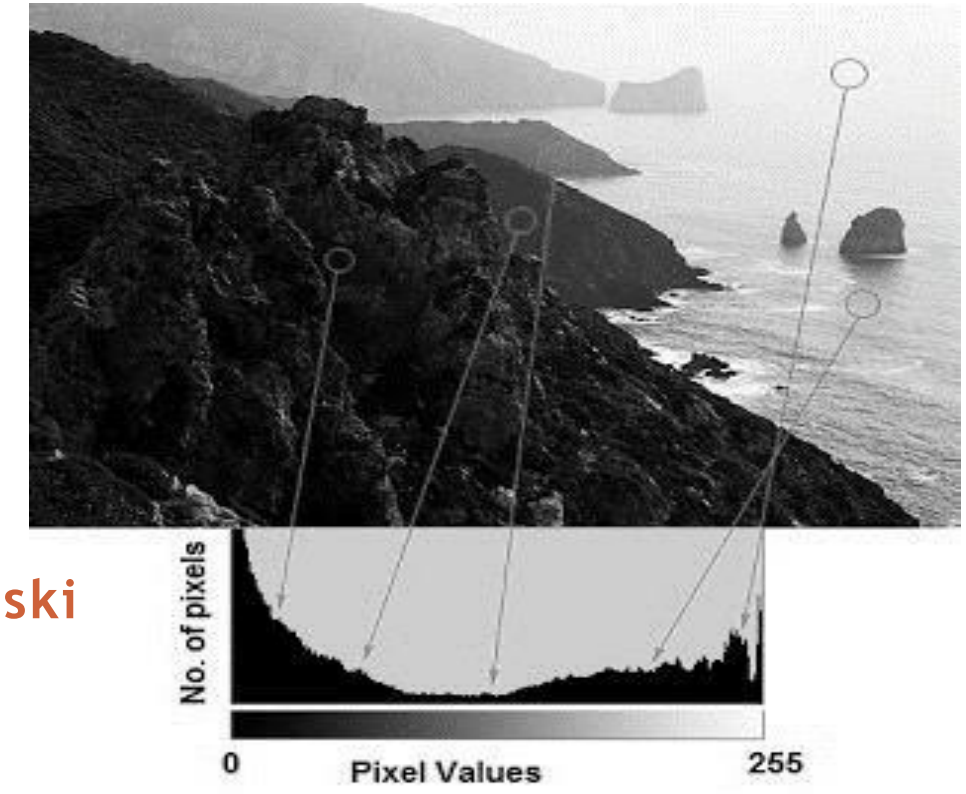


217	255	255	255
255	191	255	255
255	255	127	255
255	255	255	0

Histogram

Histogram **boja** je

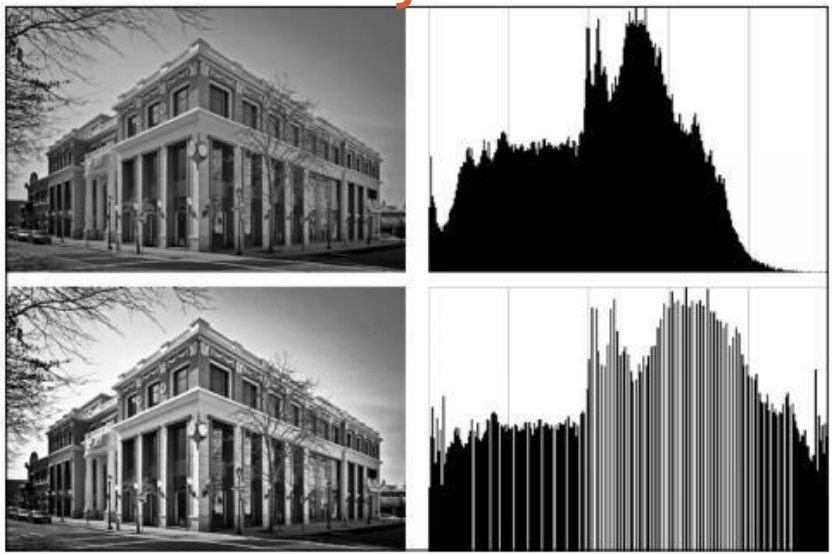
predstavljanje raspodela boja slike



Monohromatski histogram

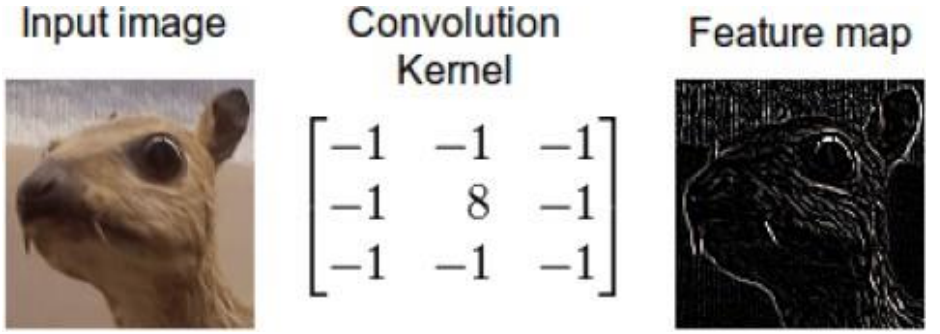
Histogram boja

Ekvalizacija historama



Filteri

Većina operacija na slikama se vrse koristeci **filtere**



- **Operacija filtriranja** je **matematicka funkcija** se koja se primenjuje na piksele slike
- Filter je predstavljen pomocu matrice, koja se naziva **kernel**
- U zavisnoti od velicina kernela ($3 \times 3, 5 \times 5, 7 \times 7$, ...), funkcija filtriranja ce obuhvatiti jedan piksel i njegove **susede**

Sadrzaj

Konvolucija

Konvolucija je postupak u kojem se svaki element slike sabira sa svojim lokalnim susedima, uz tezinske faktore odredjene kernelom.

$$y[1,1] = \sum_j \sum_i x[i,j] \cdot k[1-i, 1-j]$$

Input image $x[\cdot,\cdot]$

1	2	3
4	5	6
7	8	9

Kernel $k[\cdot,\cdot]$

1	2	1
0	0	0
-1	-2	-1

STEP 5

1	2	1
0	0	0
-1	-2	-1

Output $y[\cdot,\cdot]$

-13	-20	-17
-18	-24	

$$1 \cdot \mathbf{1} + 2 \cdot \mathbf{2} + 3 \cdot \mathbf{1} + 4 \cdot \mathbf{0} + 5 \cdot \mathbf{0} + 6 \cdot \mathbf{0} + 7 \cdot (\mathbf{-1}) + 8 \cdot (\mathbf{-2}) + 9 \cdot (\mathbf{-1})$$

Konvolucija

Konvolucija je postupak u kojem se svaki element slike sabira sa svojim lokalnim susedima, uz tezinske faktore odredjene kernelom.

$$y[0,0] = \sum_j \sum_i x[i,j] \cdot k[0-i, 0-j]$$

Ulazna slika $x[\cdot,\cdot]$

1	2	3
4	5	6
7	8	9

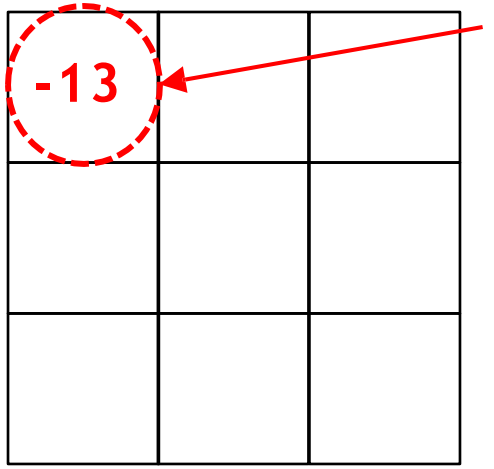
Kernel $k[\cdot,\cdot]$

1	2	1
0	0	0
-1	-2	-1

Korak 1

1	2	1	
0	0	0	3
-1	-2	-1	6
	7	8	9

Izlaz $y[\cdot,\cdot]$



$$0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 + 0 \cdot 0 + 1 \cdot 0 + 2 \cdot 0 + 0 \cdot (-1) + 4 \cdot (-2) + 5 \cdot (-1)$$

Konvolucija

Konvolucija je postupak u kojem se svaki element slike sabira sa svojim lokalnim susedima, uz tezinske faktore odredjene kernelom.

Ulazna slika $x[\cdot,\cdot]$

1	2	3
4	5	6
7	8	9

Kernel $k[\cdot,\cdot]$

1	2	1
0	0	0
-1	-2	-1

$$y[2,2] = \sum_j \sum_i x[i,j] \cdot k[2-i, 2-j]$$

Korak 9

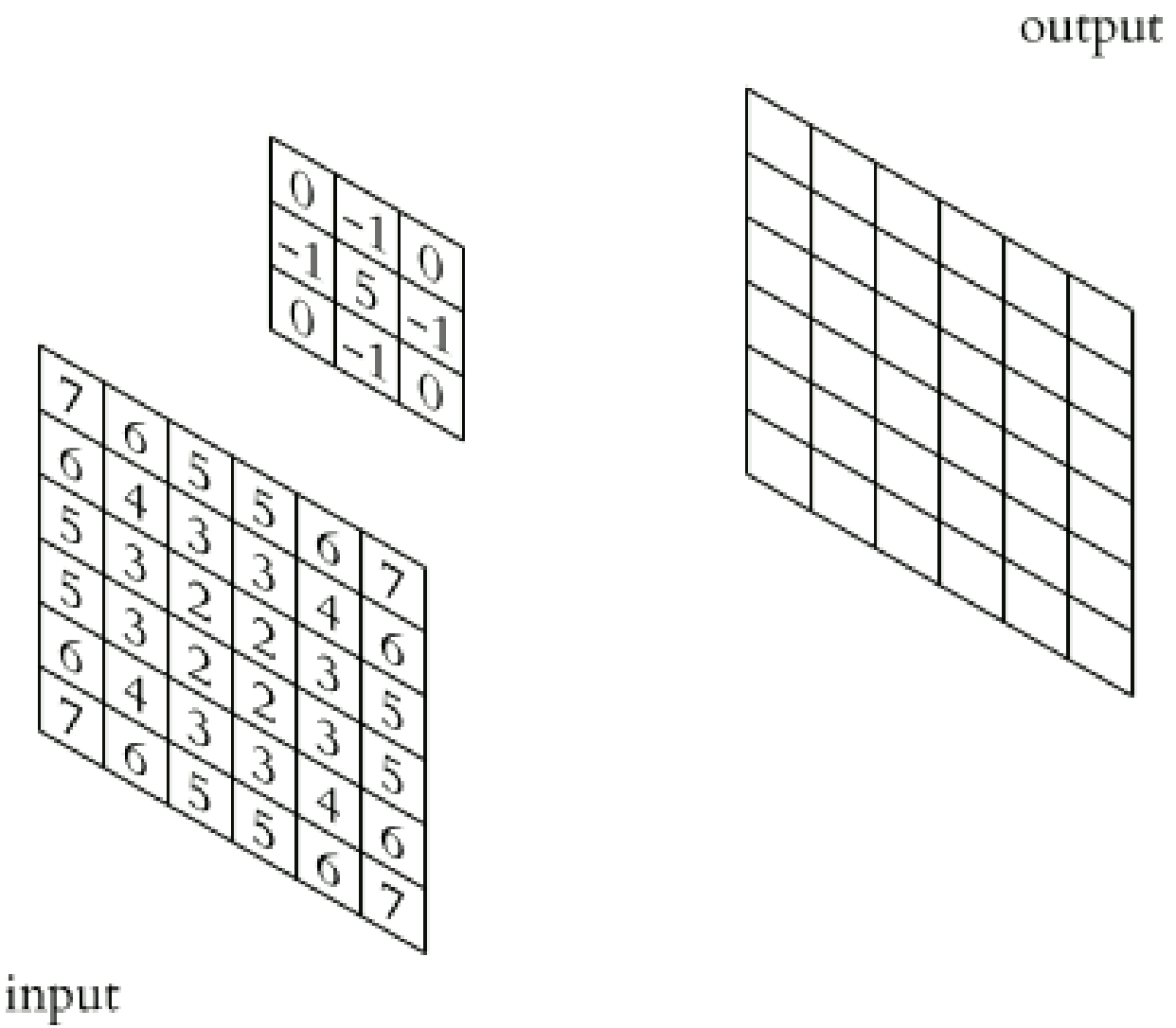
1	2	3	
4	1	2	1
7	0	0	0
	-1	-2	-1

Izlaz $y[\cdot,\cdot]$

-13	-20	-17
-18	-24	-18
13	20	17

$$5 \cdot \mathbf{1} + 6 \cdot \mathbf{2} + 0 \cdot \mathbf{1} + 8 \cdot \mathbf{0} + 9 \cdot \mathbf{0} + 0 \cdot \mathbf{0} + 0 \cdot (\mathbf{-1}) + 0 \cdot (\mathbf{-2}) + 0 \cdot (\mathbf{-1})$$

Konvolucija



https://it.m.wikipedia.org/wiki/File:3D_Convolution_Animation.gif

Kernel primeri

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Poostreno

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Detekcija-ivica

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Blur/pomeraj proseka

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Kernel primeri

Ivice

-1	-1	-1
-1	8	-1
-1	-1	-1

45° Linije

-1	-1	2
-1	2	-1
2	-1	-1

Horizontalne linije

-1	-1	-1
2	2	2
-1	-1	-1

Vertikalne linije

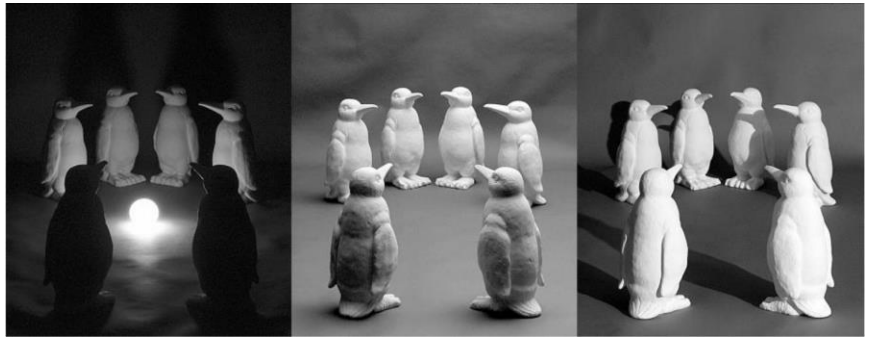
-1	2	-1
-1	2	-1
-1	2	-1

Sadrzaj

Klasicni pristupi

Zadaci racunarskog vida moraju biti **robusni** na vise vrsta poremecaja:

Razlicite tacke gledista



Razlicito osvetljenje

Deformacije

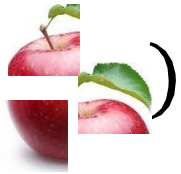
Razliciti oblici istog objekta

Kako se identifikuje objekat?

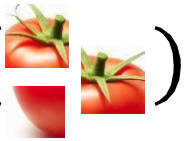
Ideja je izvuci **karakteristike(features)**, odnosno neke vazne osobine slike

OBJEKAT

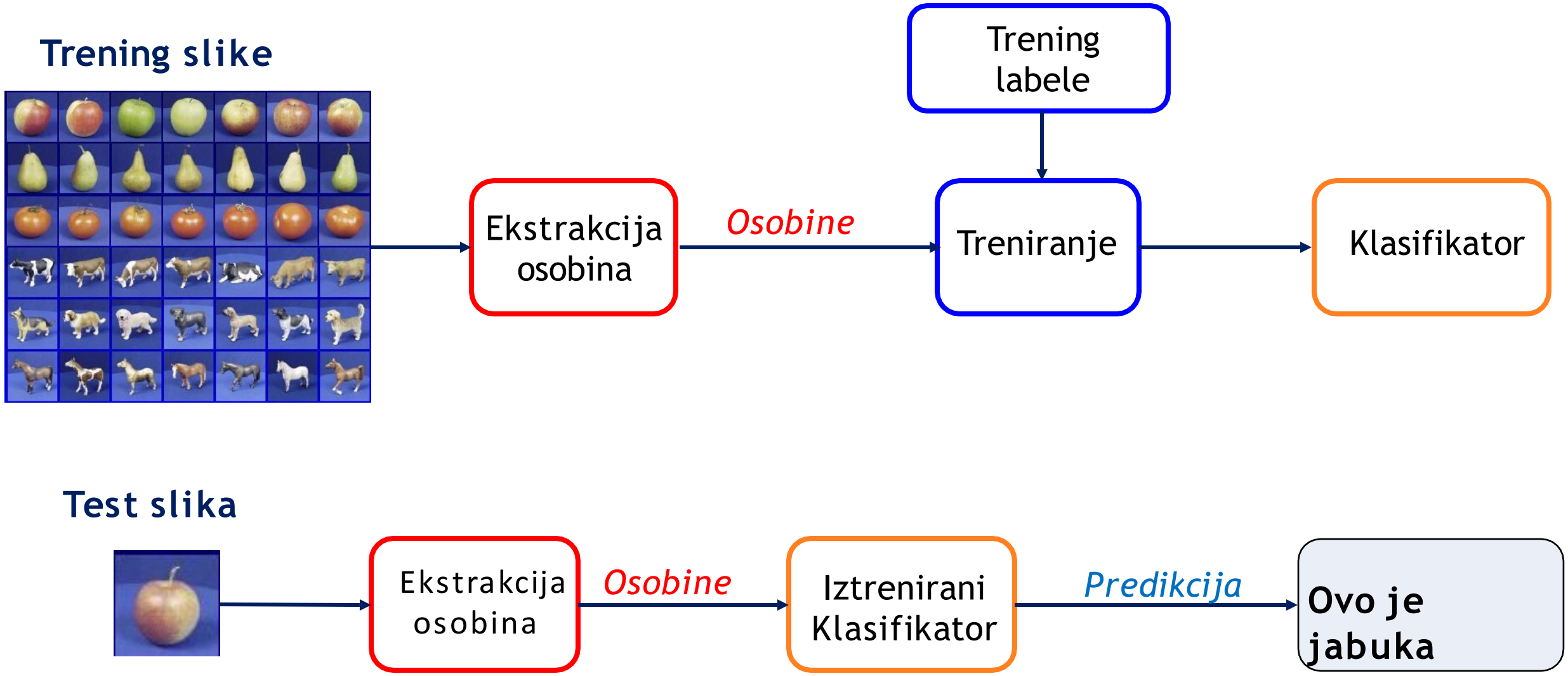
“BAG OF FEATURES”

$f($  $) =$ **jabuka**

$f($  $) =$ **krava**

$f($  $) =$ **paradajz**

Klasfikacijski "pipeline"



Osobine slika

U Racunarskom Vidu, osobina/osobine je deo informacije kojiije releventan za **resenje odredjenog problema** (npr. pri detekciji lica, prisustvo dva oka je dobra osobina)

Tri hijerarhijske kategorije osobina:

Low-level osobine

- Boje
- Ivice
- "Blob"
- Uglovi

Mid-level osobine

- Scale-invariant features
- SIFT
- SURF
- SVD

High-level osobine

- Histograms of gradients (HOG)
- Regionski descriptori
- Haar osobine

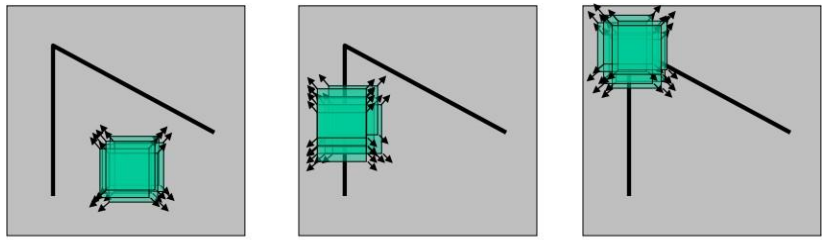
Low-level osobine

Ivice i uglovi: posto proces klasifikacie slika ukljucuje koriscenje ivica i uglova, postoji mnogo metoda za njihovo pronalazenje:

Canny edge detector

Koristi algoritam sa 5 koraka, koji ukljucuje neke filtere (gradijente) kako bi izracunao sve ivice na slici

Harris corner detection



Pomerajuci prozor se skrece preko slike i racuna "the Hessian". Uglovi se procenjuju uzimanjem sopstvene vrednosti svake matrice

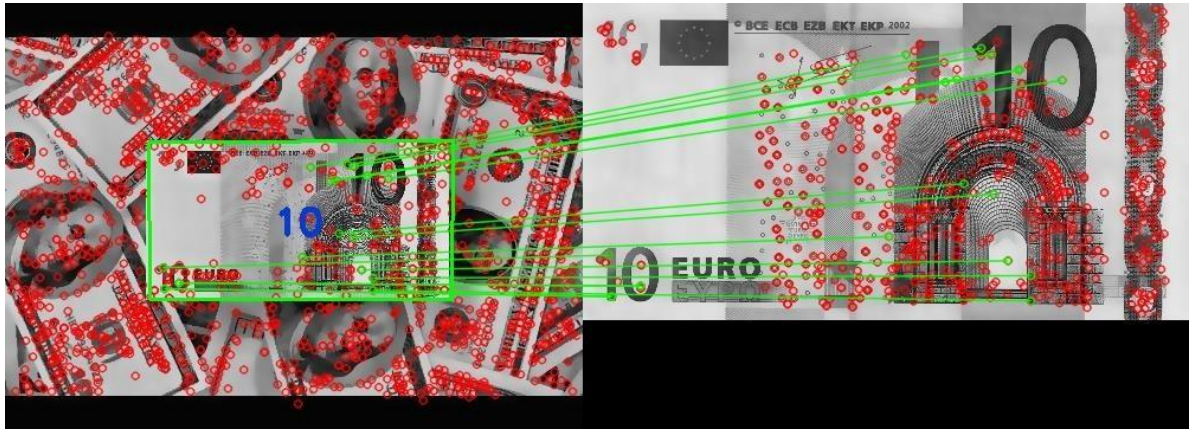
Hough transform

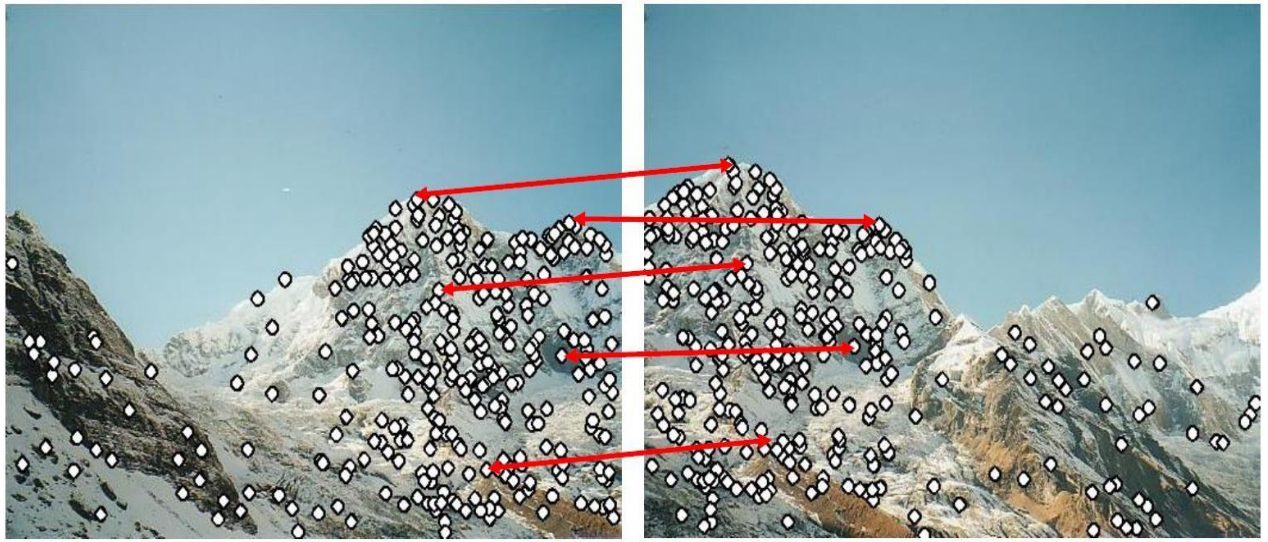
Prvobitno osmisljen za otkrivanje linija na slikama, prosiren je kako bi detektovao krugove i elipse

Mid-level osobine

SIFT - Scale Invariant Feature Transform [1999]

To je algoritam koji je sposoban da detektuje i opisuje osobine na slici pri razlicitim **skalama i rotacijama**

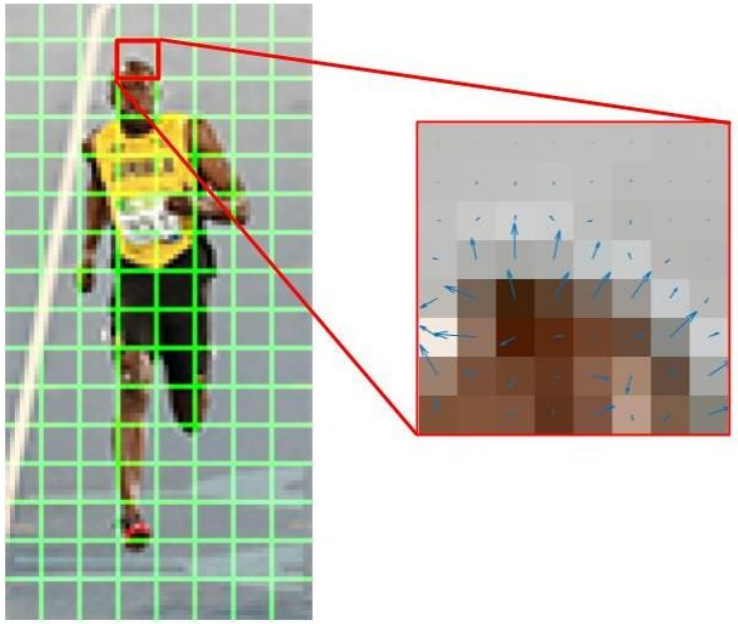




High-level osobine

HOG - Histogram of Oriented Gradients [2005]

- Broji pojavljivanja orijentacija gradijenata u lokalizovanim delovima slike
- Distribucija (histograma) pravaca gradijenata (orijentisani gradijenti) koristi se kao osobina – *velicina gradijenata je velika oko ivica i uglova (regioni naglih promena intenziteta)*

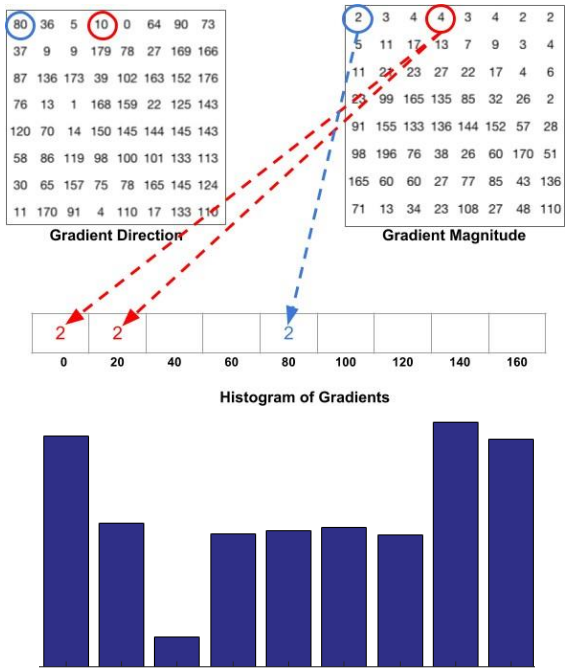


2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

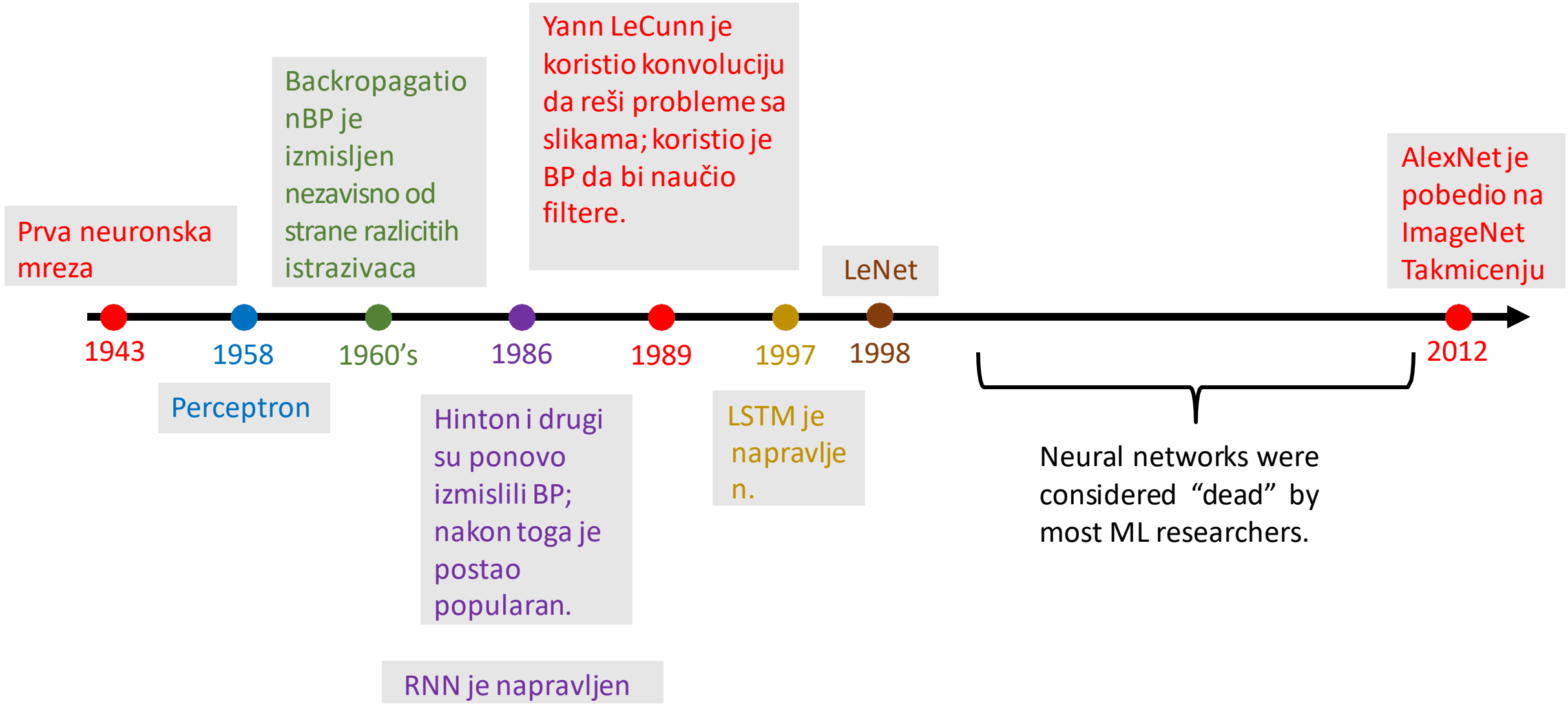
Gradient Direction



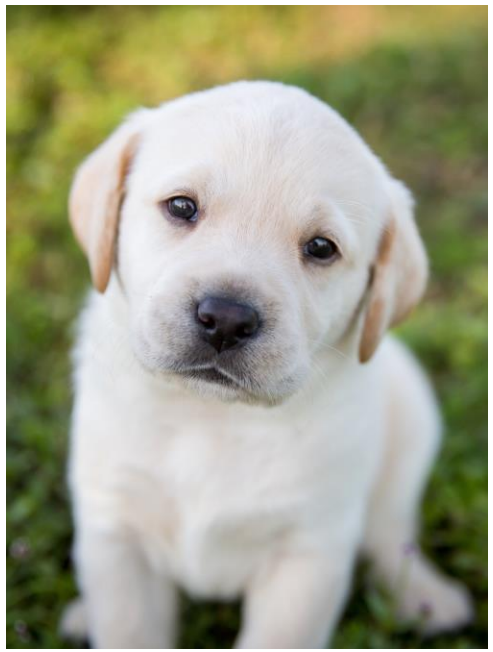
9 celija koje odgovaraju uglovima

Sadrzaj

Istorija neuronskih mreza (CNN)



Klasican pristup

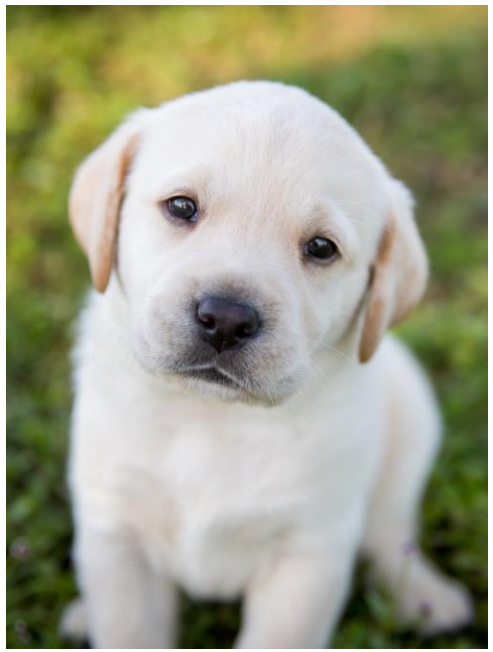


SIFT osobine

HOG osobine

LBP osobine

Klasican pristup



SIFT osobine

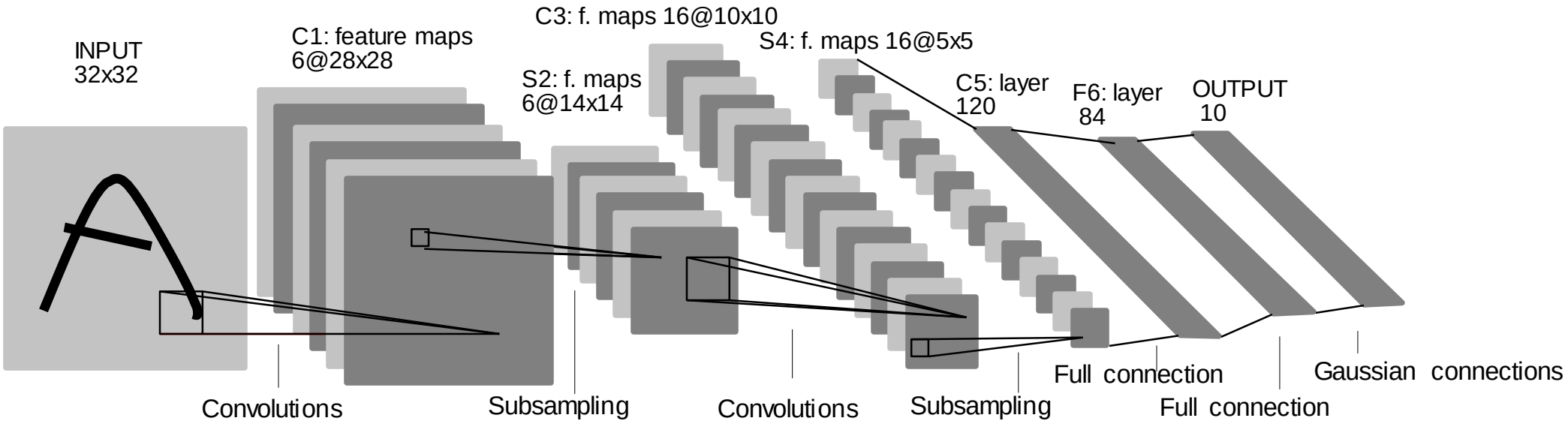
HOG osobine

LBP osobine

Linearni Model

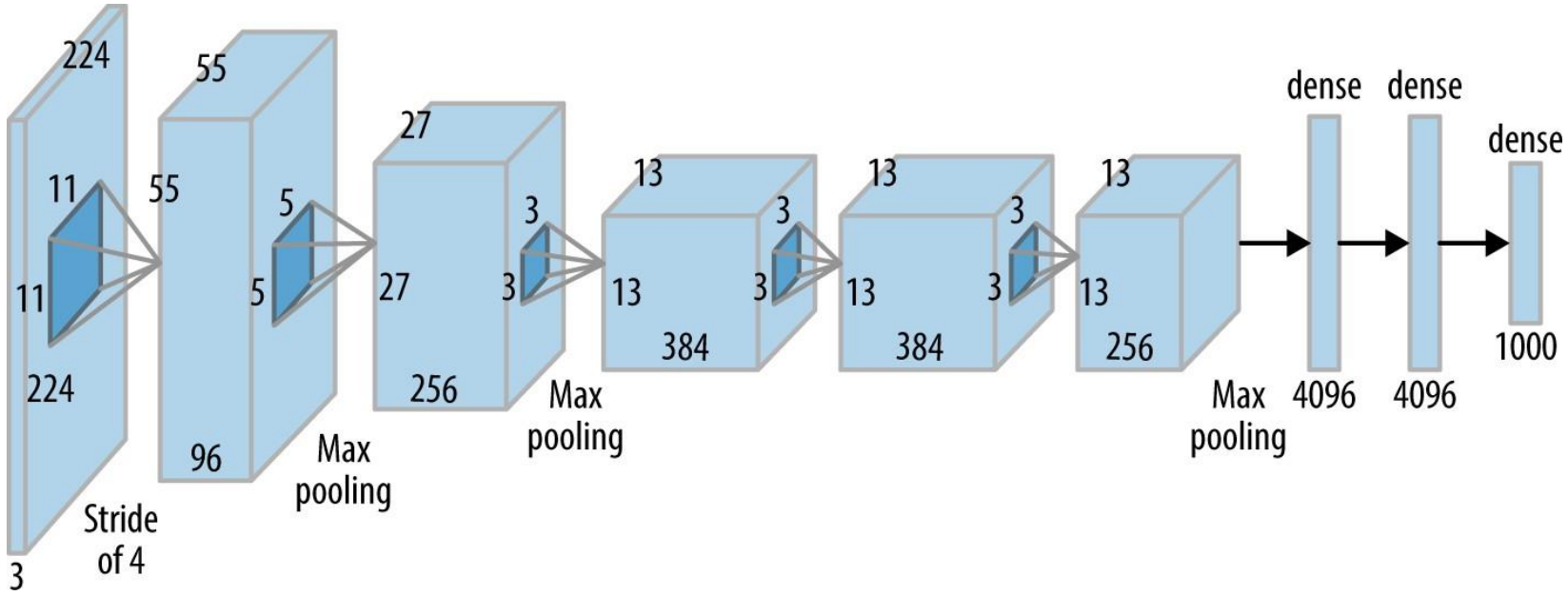
“pas”

LeNet-5 [Yan LeCun 1998]

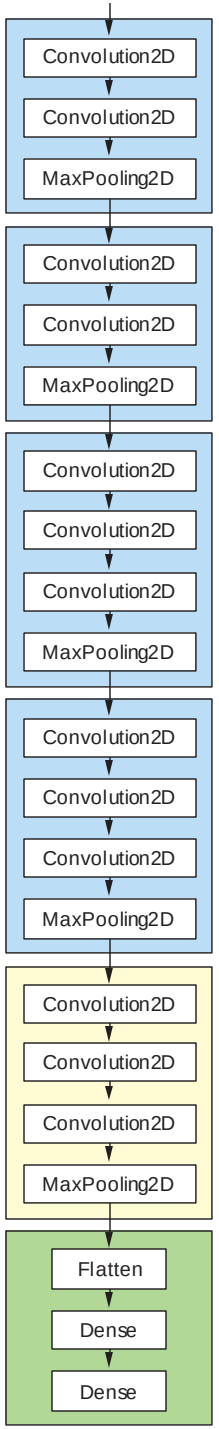


- Slojevi: 2 Conv + 2 FC layers.
- Clanak: Gradient-based learning applied to document recognition

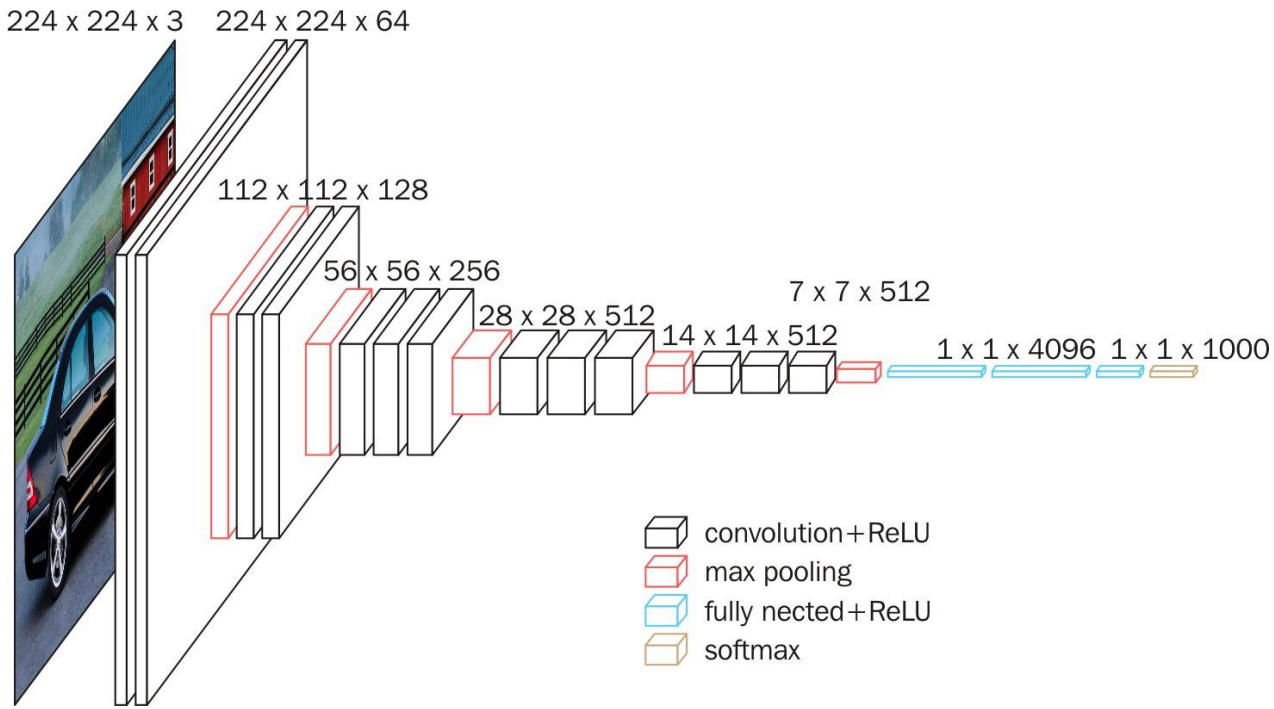
AlexNet [Alex Krizhevsky 2012]



- Slojevi: 5 Conv + 3 FC layers.
- Broj obučavajućih parametara: 60m.
- Clanak: ImageNet Classification with Deep Convolutional Neural Networks
- Keras implementacija: http://dandxy89.github.io/ImageModels/



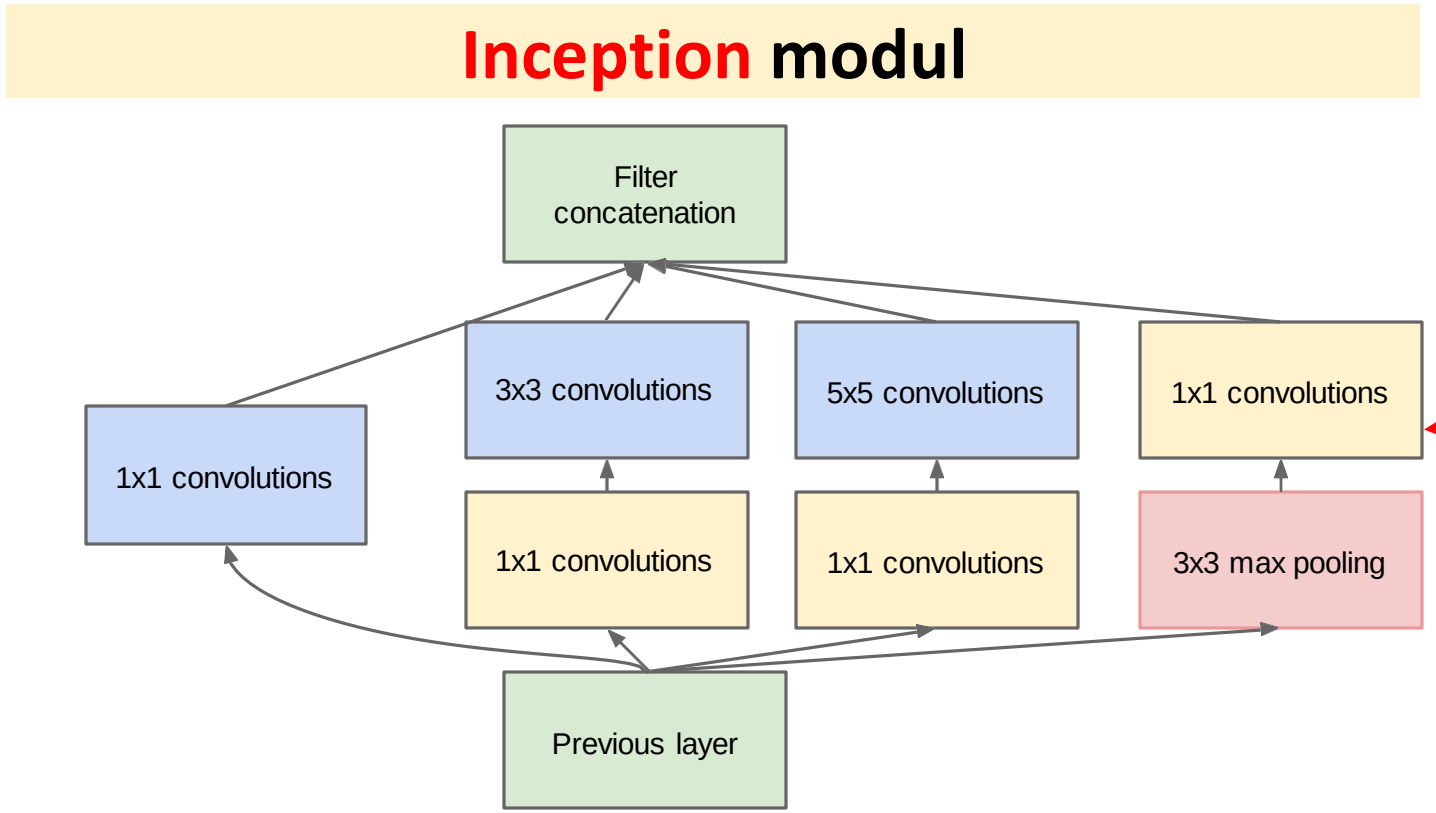
VGG16 [Simonyan & Zisserman 2014]

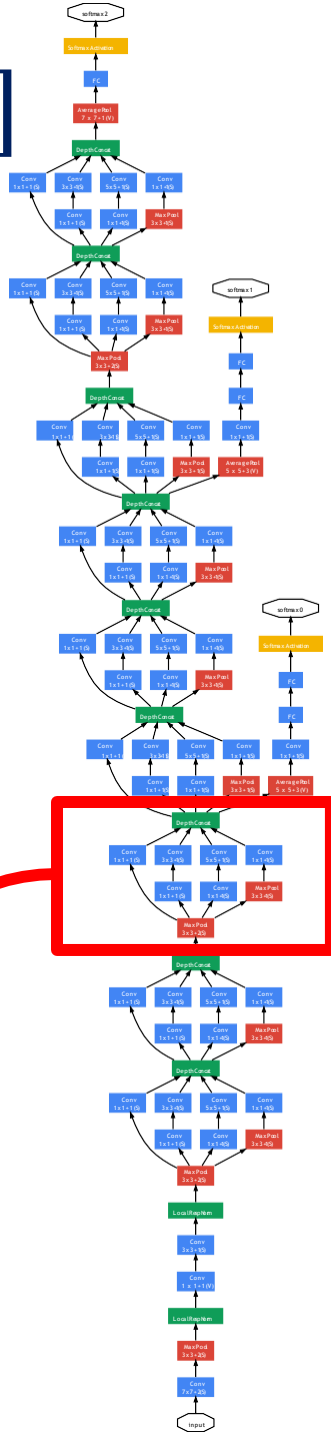


- Slojevi: 13 Conv + 2 FC layers.
- Broj obučavajućih parametara: 138m.
- Clanak: [Very Deep Convolutional Networks for Large-Scale Image Recognition]

GoogLeNet/Inception [Szegedy i dr. 2015]

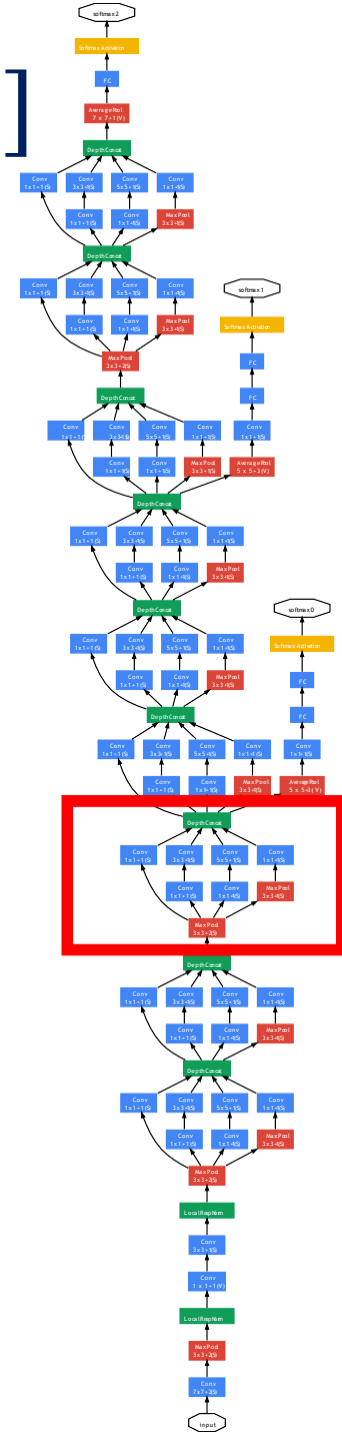
- Magacin "Inception" modula.
- Samo 5m parametara. (VGG16 ima 138m parametara!)





GoogLeNet/Inception [Szegedy i dr. 2015]

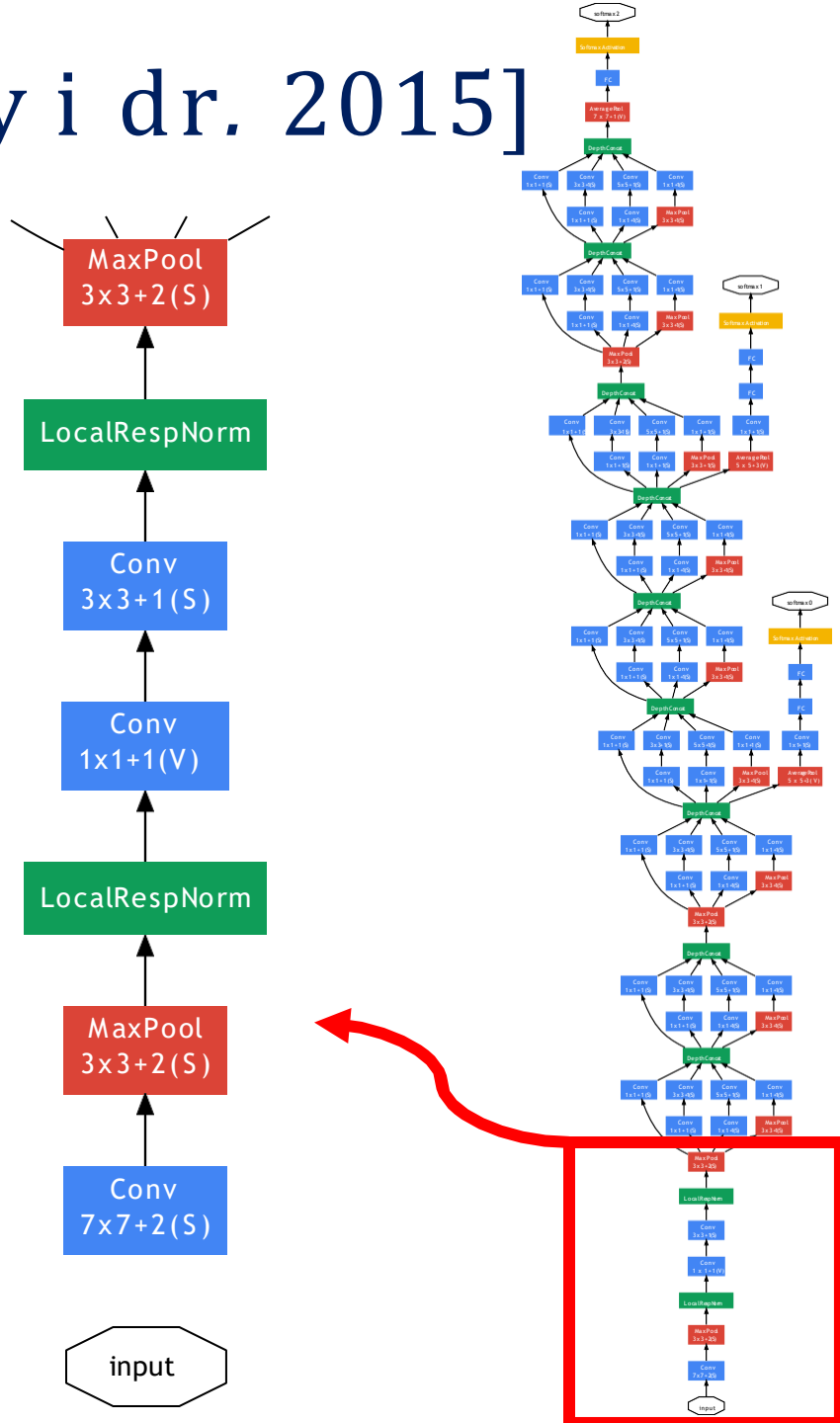
- Magacin “Inception” modula.
- Samo 5m parametara. (VGG16 ima 138m parametara!)



GoogLeNet/Inception [Szegedy i dr. 2015]

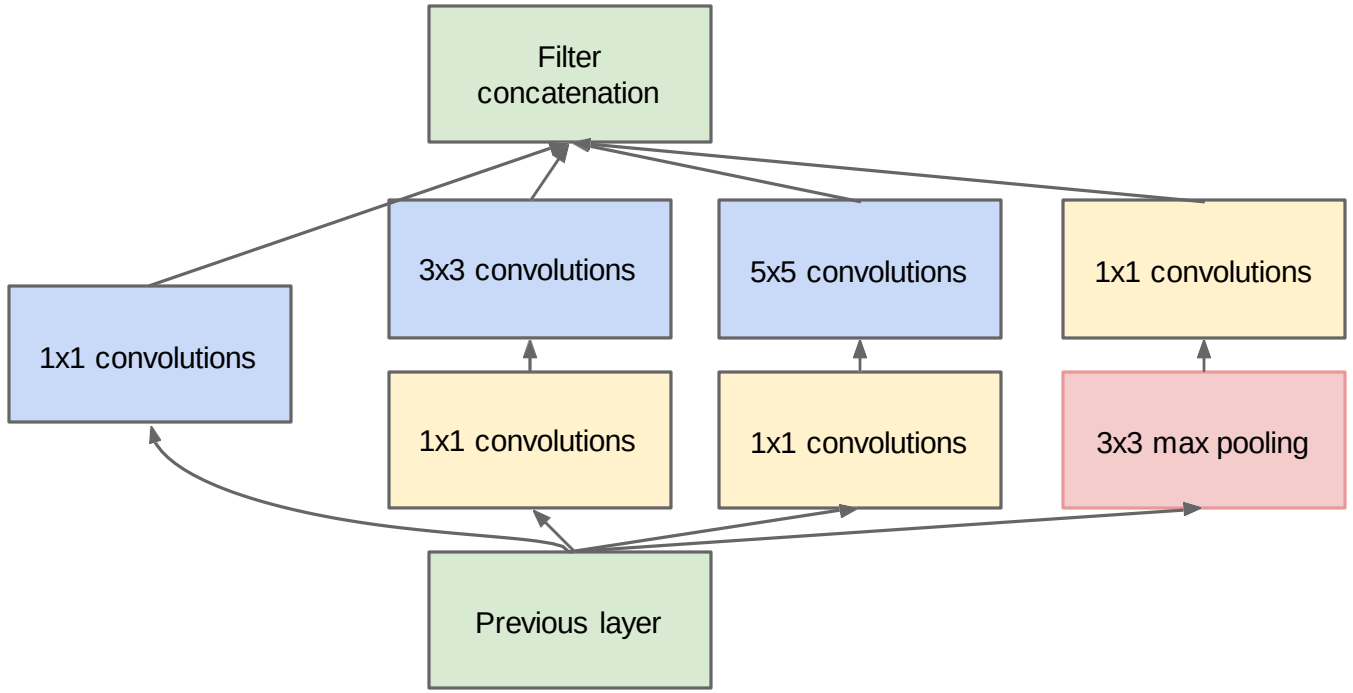
Donji slojevi

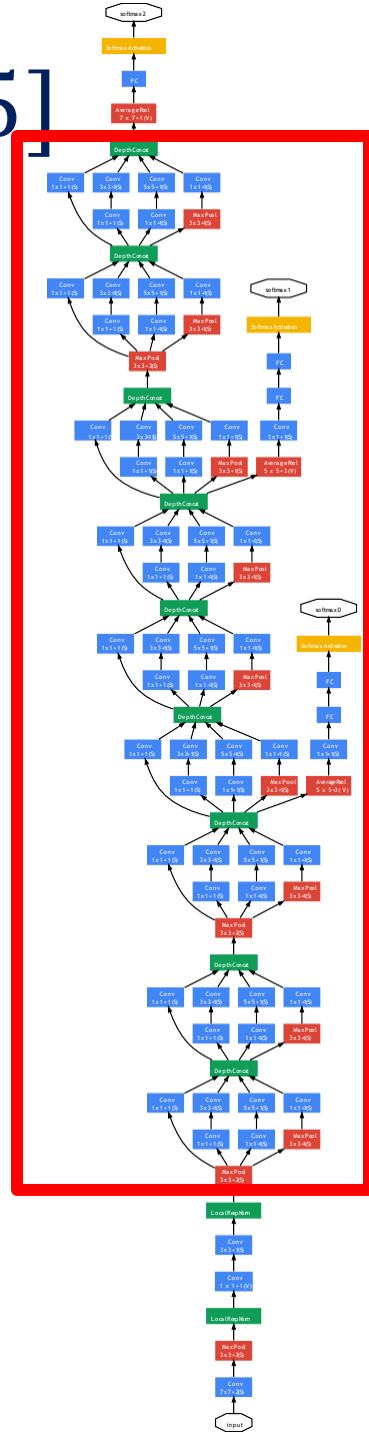
- Jednostavan ConvNet.
- 3 Conv + 2 Pooling + 2 Normalization-a sloja.



GoogLeNet/Inception [Szegedy i dr. 2015]

Magacin 9 Inception modula



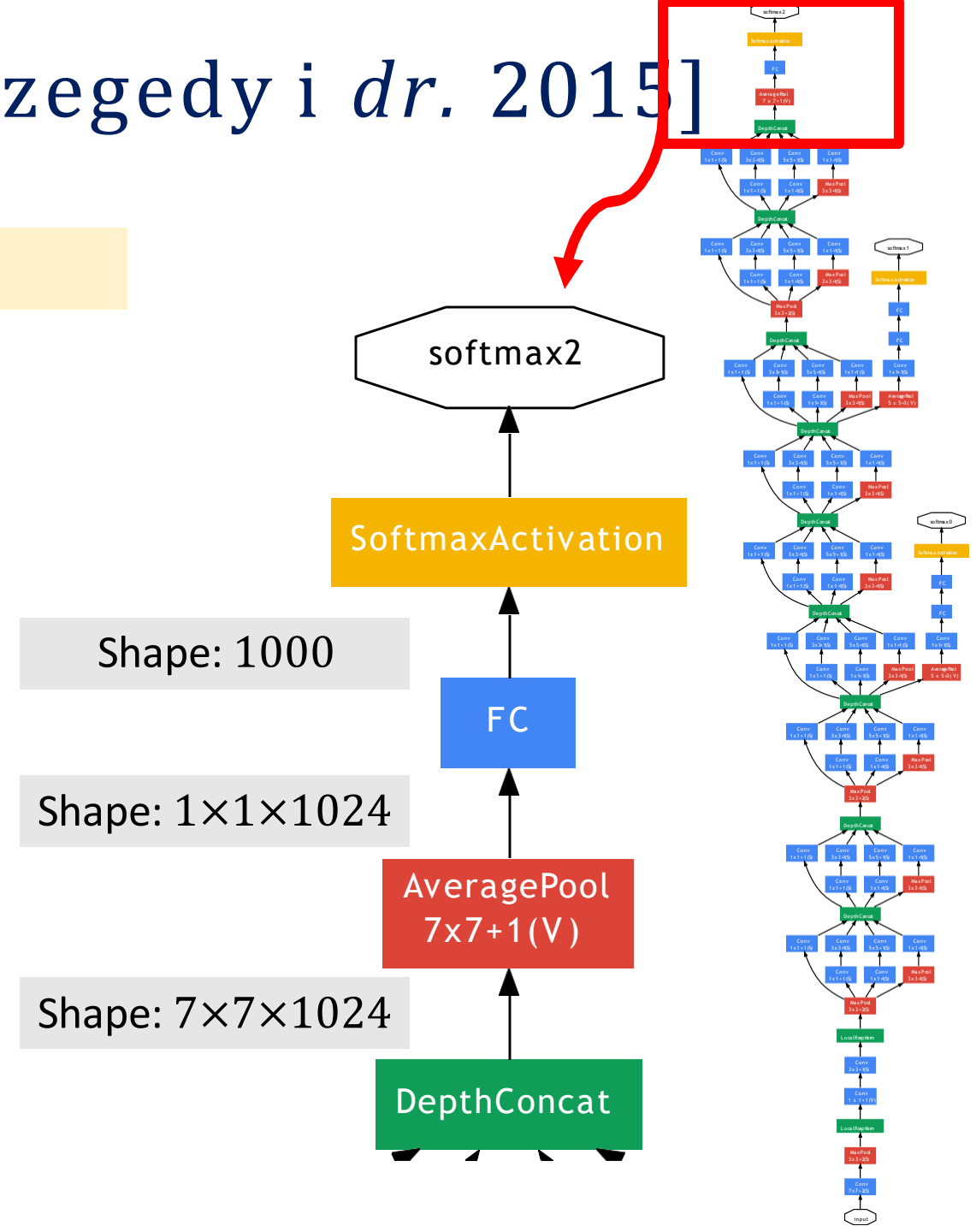


GoogLeNet/Inception [Szegedy i *dr.* 2015]

Izlazni slojevi

- Koristi AveragePool umesto Flatten
- AveragePool sloj sluzi za umanjenje broja parametera

Pitanje: Ako se AveragePool zameni sa Flatten, koliki bi broj parametara bio unutar FC sloja?



GoogLeNet/Inception [Szegedy *et al.* 2015]

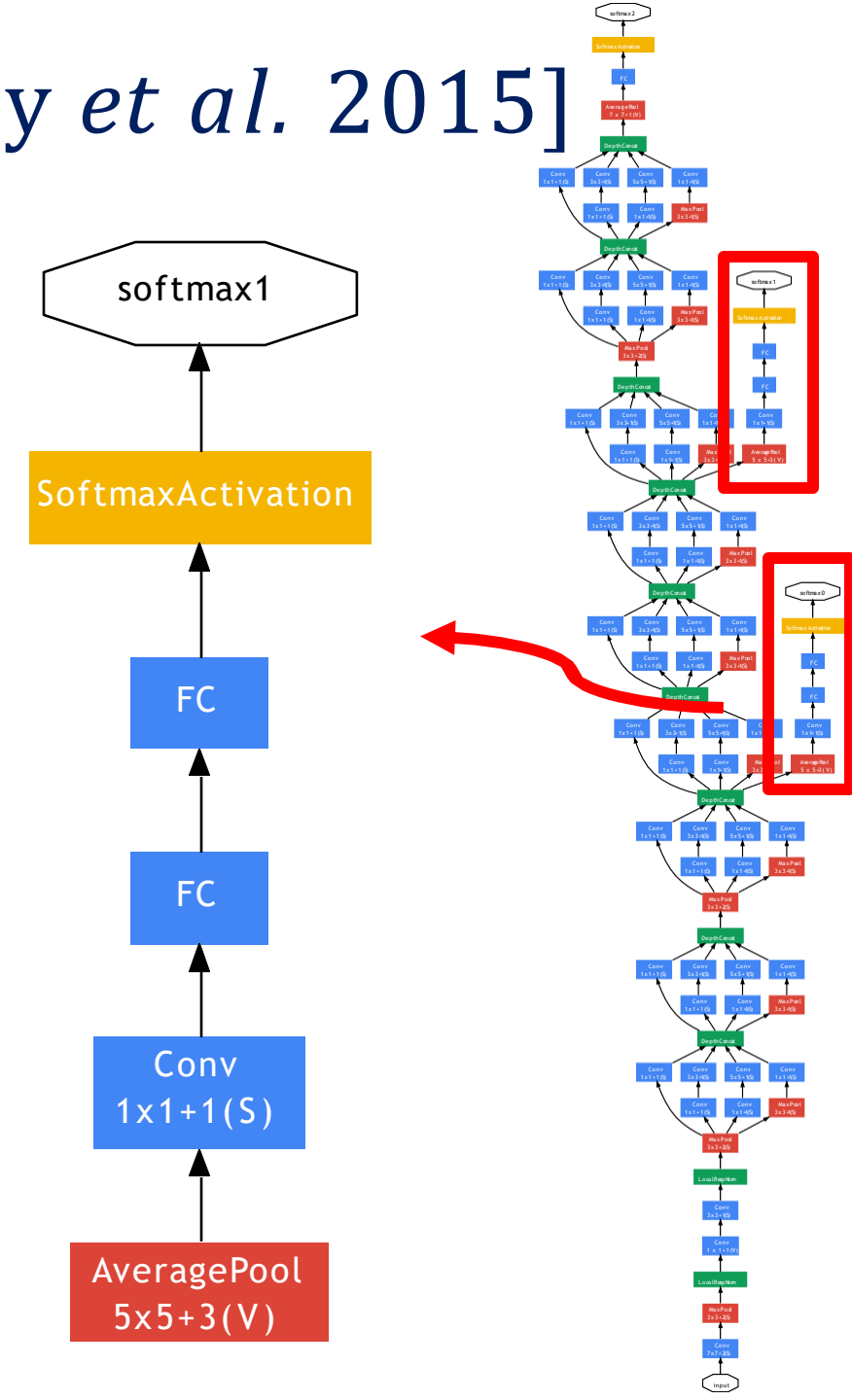
Pomocni izlazi

Tokom Treniranja:

- Ubacuje dodatni gradijent u nize slojeve
- Olaksava optimizaciju

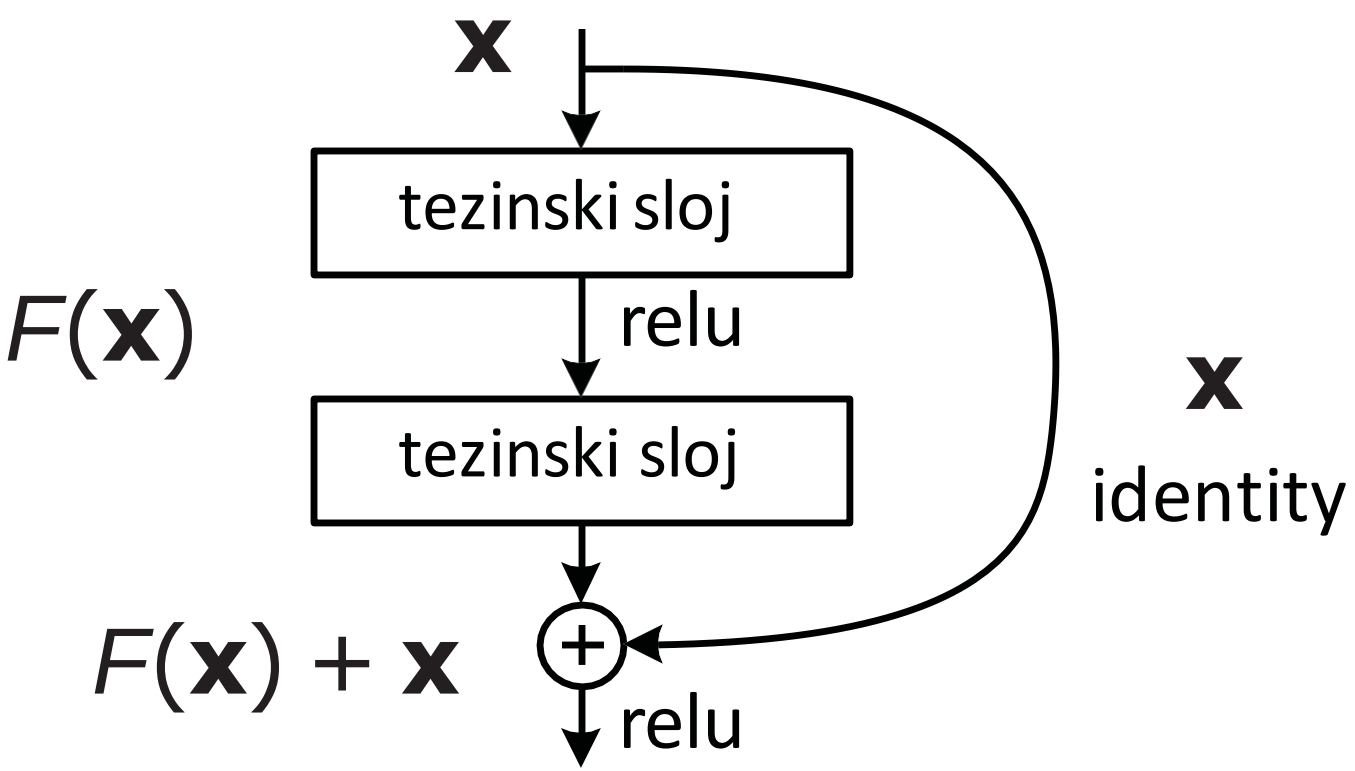
Tokom Testiranja:

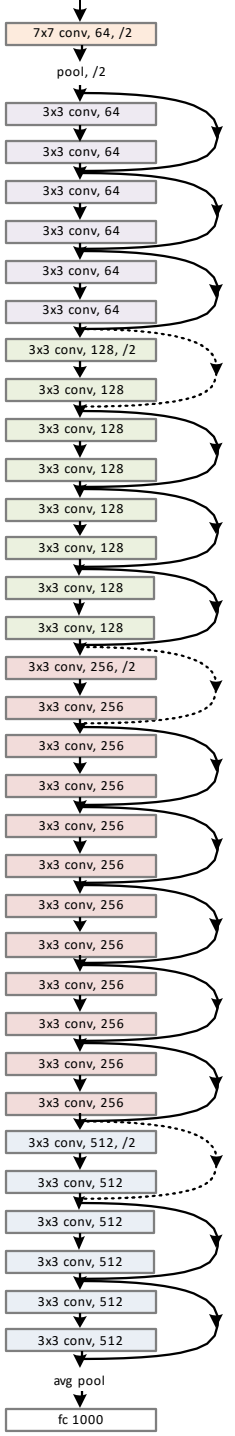
- Onemogucuje pomocne izlaze.



ResNet [2015]

Blok od ResNet-a





CNN Arhitekture

- LeNet-5: 2 Conv + 2 FC sloja.
- AlexNet: 5 Conv + 3 FC sloja.
- VGG16: 13 Conv + 2 FC sloja.

} klasicne arhitekture (sequential)

CNN Arhitekture

- LeNet-5: 2 Conv + 2 FC sloja.
- AlexNet: 5 Conv + 3 FC sloja.
- VGG16: 13 Conv + 2 FC sloja.

klasicne arhitekture (sequential)

- Inception: 21 Conv + 1 FC sloja.
- ResNet: Cak do 151 Conv + 1 FC sloja.

moderne arhiteture

CNN Arhitekture

- LeNet-5: 2 Conv + 2 FC sloja.
- AlexNet: 5 Conv + 3 FC sloja.
- VGG16: 13 Conv + 2 FC sloja.

} klasicne arhitekture (sequential)

Pitanje: Mogu li klasicne arhitekture biti dublje?

Odgovor: Ne.

- Dublje mreze imaju losije trening i test greske.
- Nestajuci gradijenti.

CNN Arhitekture

- LeNet-5: 2 Conv + 2 FC sloja.
- AlexNet: 5 Conv + 3 FC sloja.
- VGG16: 13 Conv + 2 FC sloja.

klasicna arhitektura (sequential)

- Inception: 21 Conv + 1 FC layers.
- ResNet: Cak do 151 Conv + 1 FC sloja.

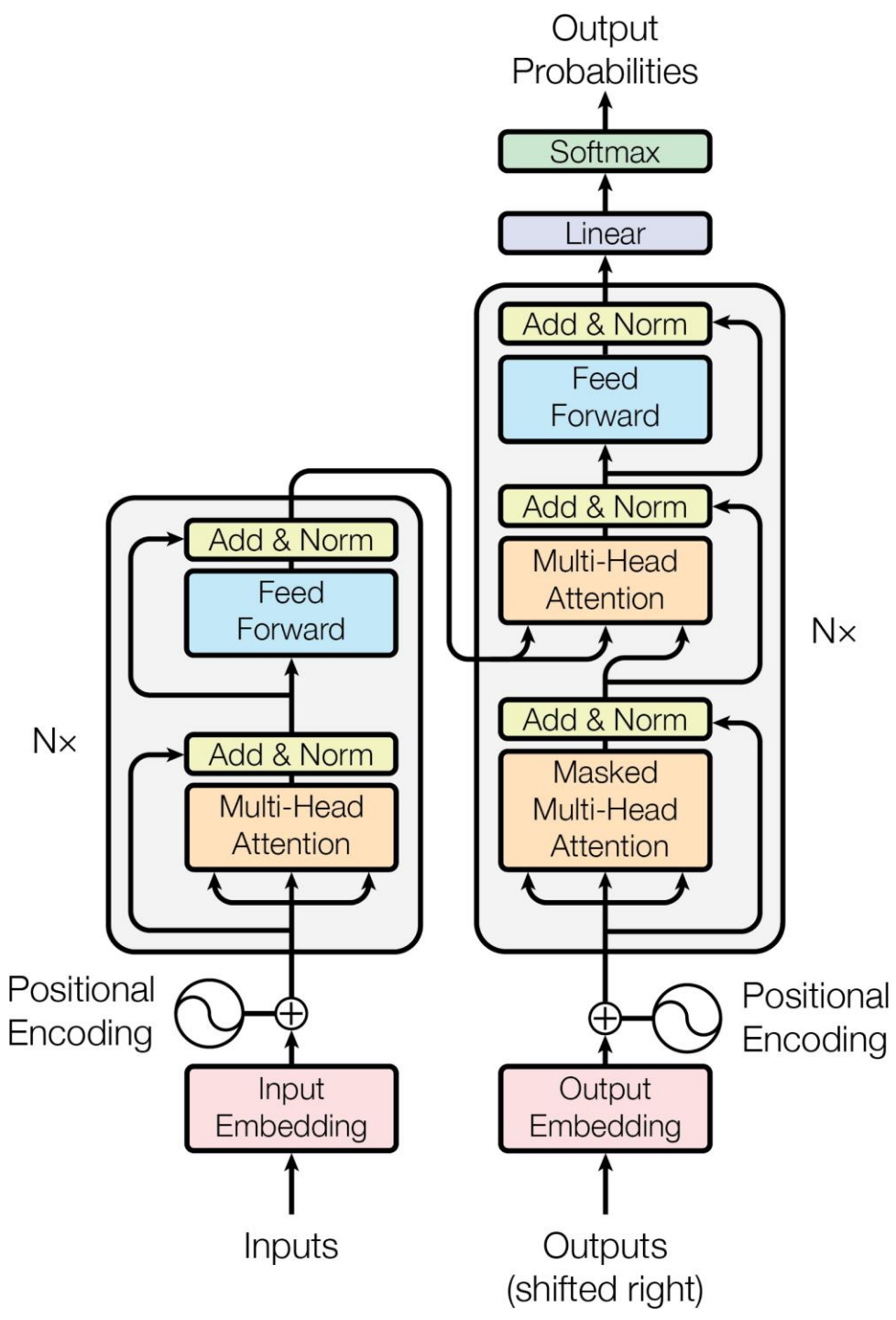
moderne arhitekture

Trikovi:

- Inception: auxiliary(pomocni) izlaz (gradient injekcija).
- ResNet: preskocna veza.

Transformer Model

- Transformer je Seq2Seq model.
- Transformer nije RNN.
- Potpuno bazirano na paznji i gustim slojevima.
- Veca tacnost nego RNNs na velikim skupovima podataka.



Attention Layer

- Proucavamo Seq2Seq model (encoder + decoder).
- Encoder-ovi ulazi su vektori $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$.
- Decoder-ovi ulazi su vektori $\mathbf{x}_1^{\cdot}, \mathbf{x}_2^{\cdot}, \cdots, \mathbf{x}_l^{\cdot}$.

Encoder's inputs:

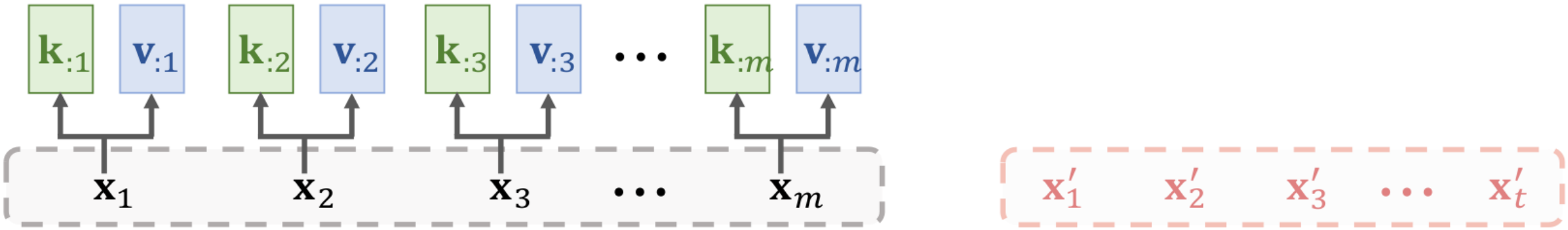
$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\cdots$ $\mathbf{x}_m$

Decoder's inputs:

$\mathbf{x}^{\cdot}_1$ $\mathbf{x}^{\cdot}_2$ $\mathbf{x}^{\cdot}_3$ $\cdots$ $\mathbf{x}^{\cdot}_:$

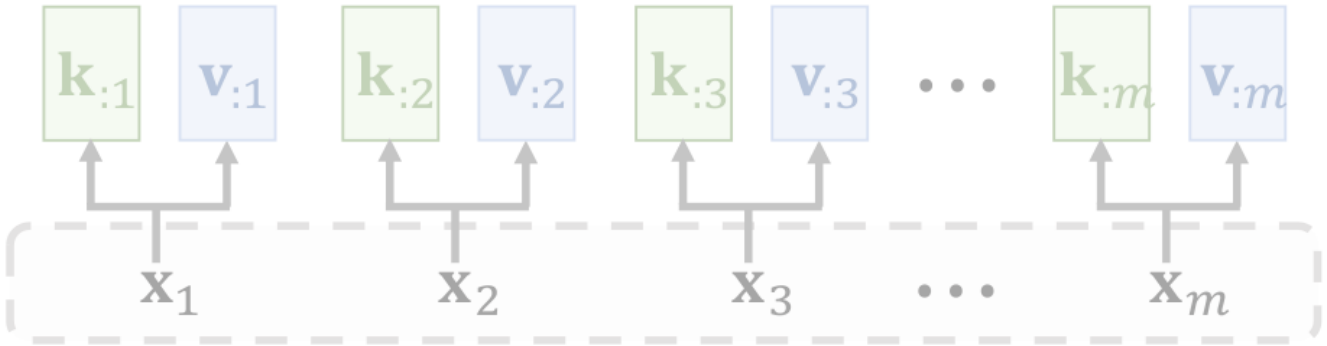
Attention Layer

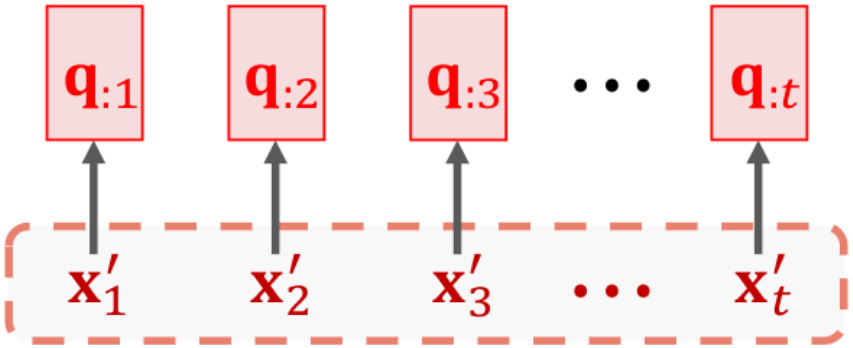
- Keys i values su bazirani na encoder ulazima $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$.
- Key(Kljuc): $\mathbf{k}_{:i} = \mathbf{W}_k \mathbf{x}_i$
- Value(Vrednost): $\mathbf{v}_{:i} = \mathbf{W}_v \mathbf{x}_i$



Attention Layer

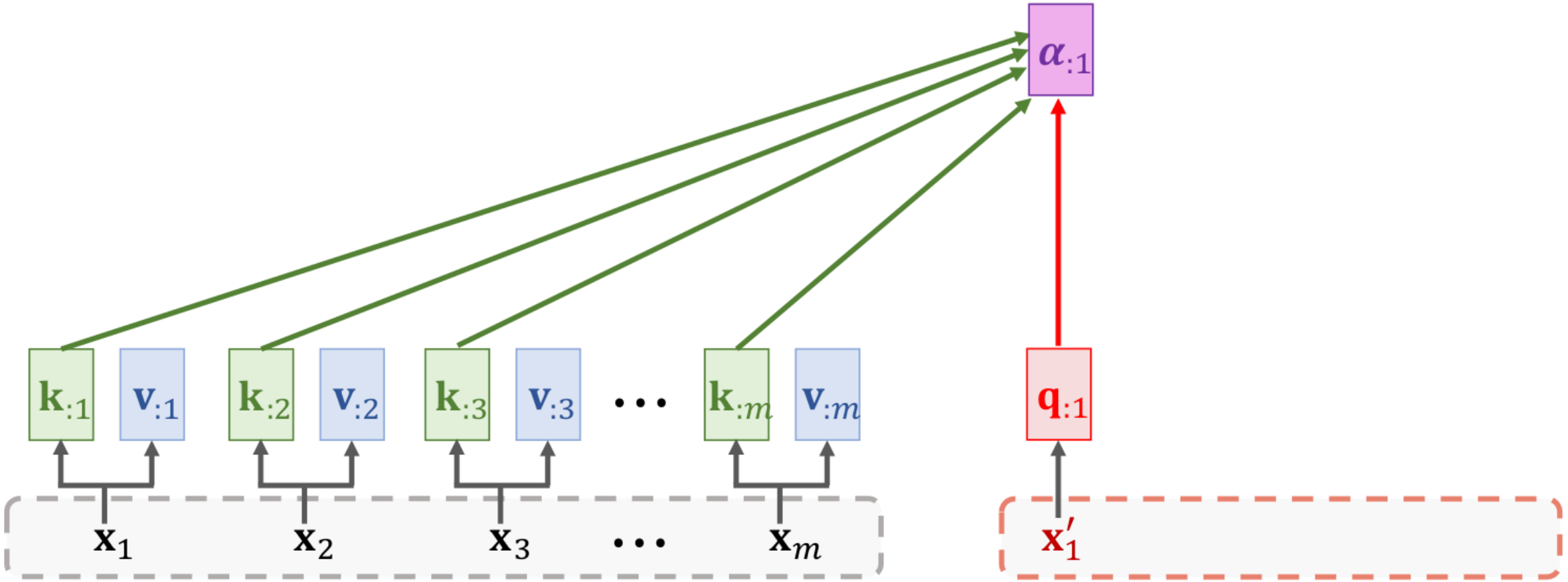
- Keys i values su bazirani na encoder ulazima $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$.
- Key(Kljuc): $\mathbf{k}_{:i} = \mathbf{W}_k \mathbf{x}_i$
- Value(Vrednost): $\mathbf{v}_{:i} = \mathbf{W}_v \mathbf{x}_i$
- Queries su bazirani na decoder izlazima $\mathbf{x}'_1, \mathbf{x}'_2, \cdots, \mathbf{x}'_t$
- Query: $\mathbf{q}_{:j} = \mathbf{W}_Q \mathbf{x}'_j$





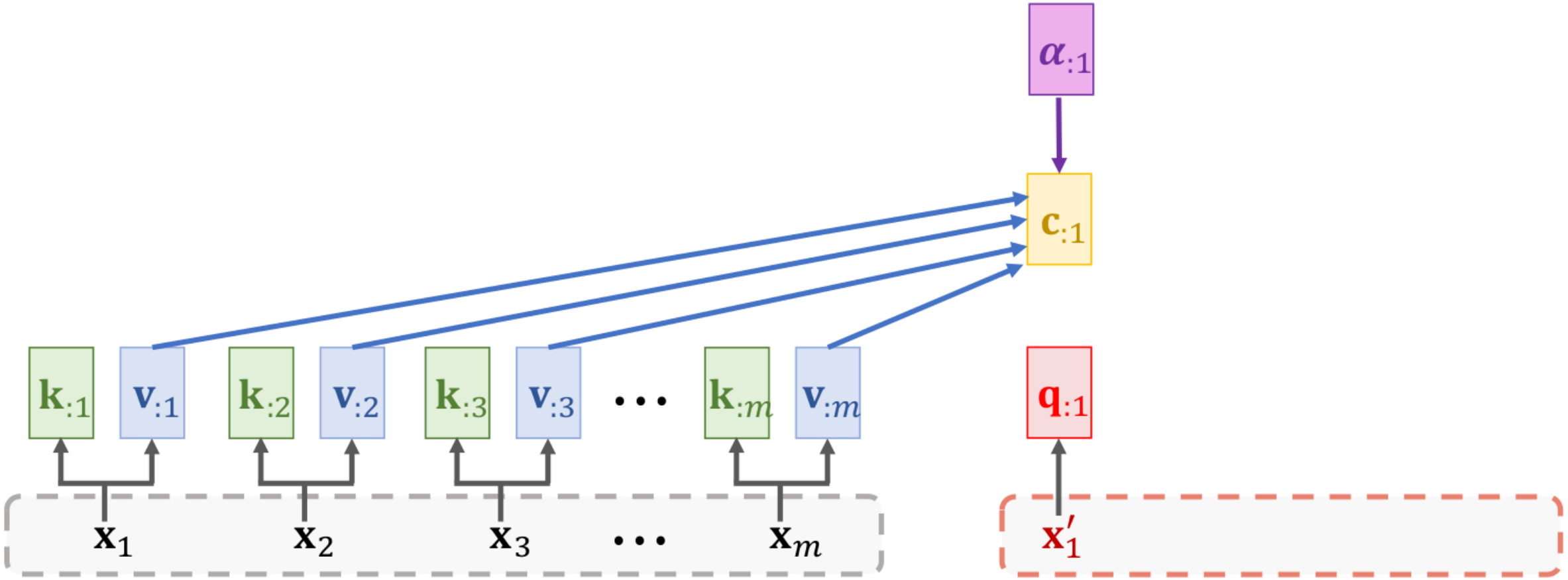
Attention Layer

- Compute weights: $\boldsymbol{\alpha}_{:1} = \text{Softmax}(\mathbf{K}^T \mathbf{q}_{:1}) \in \mathbb{R}^m$.



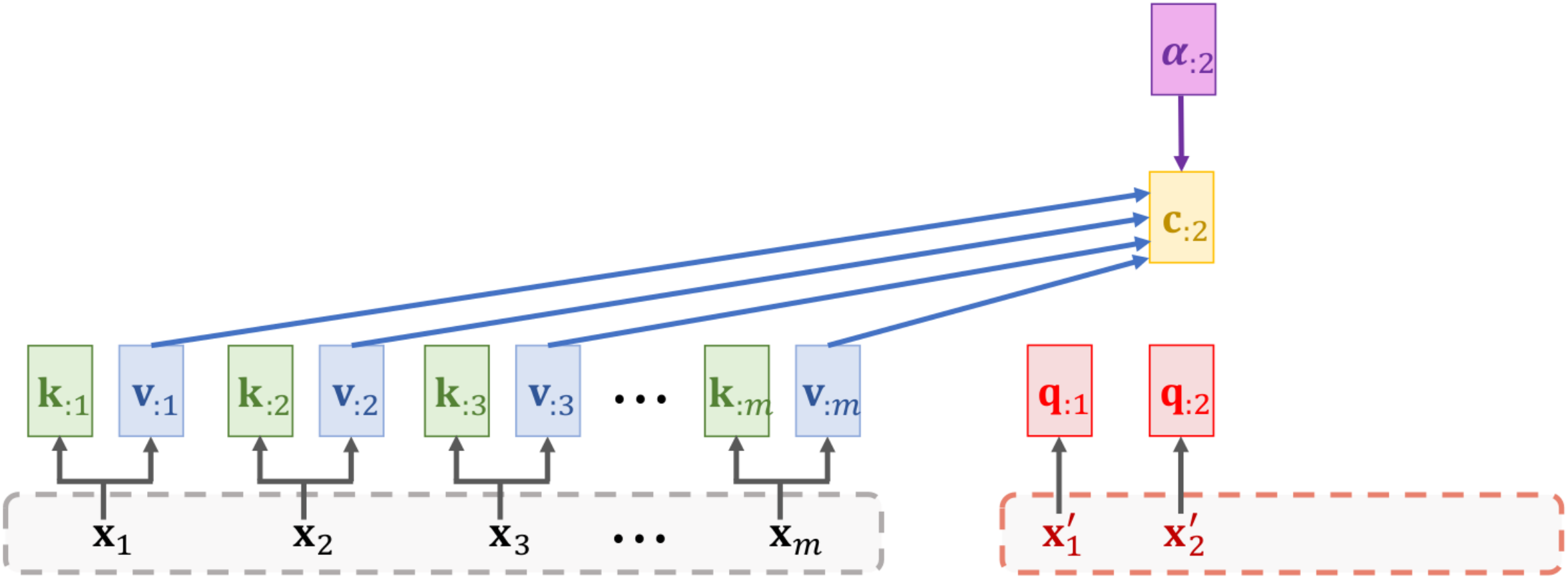
Attention Layer

- Compute context vector: $\mathbf{c}_{:1} = \alpha_{11}\mathbf{v}_{:1} + \cdots + \alpha_{m1}\mathbf{v}_{:m} = \mathbf{V}\boldsymbol{\alpha}_{:1}$.



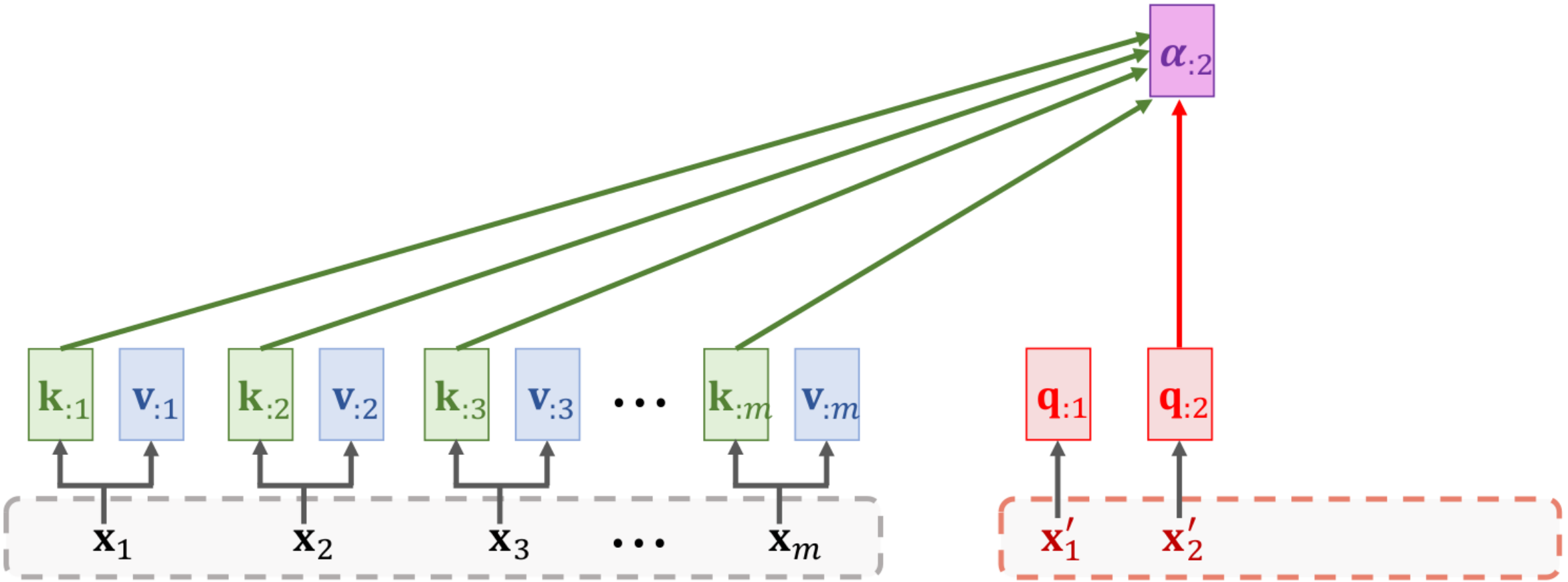
Attention Layer

- Compute context vector: $\mathbf{c}_{:2} = \alpha_{12}\mathbf{v}_{:1} + \cdots + \alpha_{m2}\mathbf{v}_{:m} = \mathbf{V}\boldsymbol{\alpha}_{:2}$.



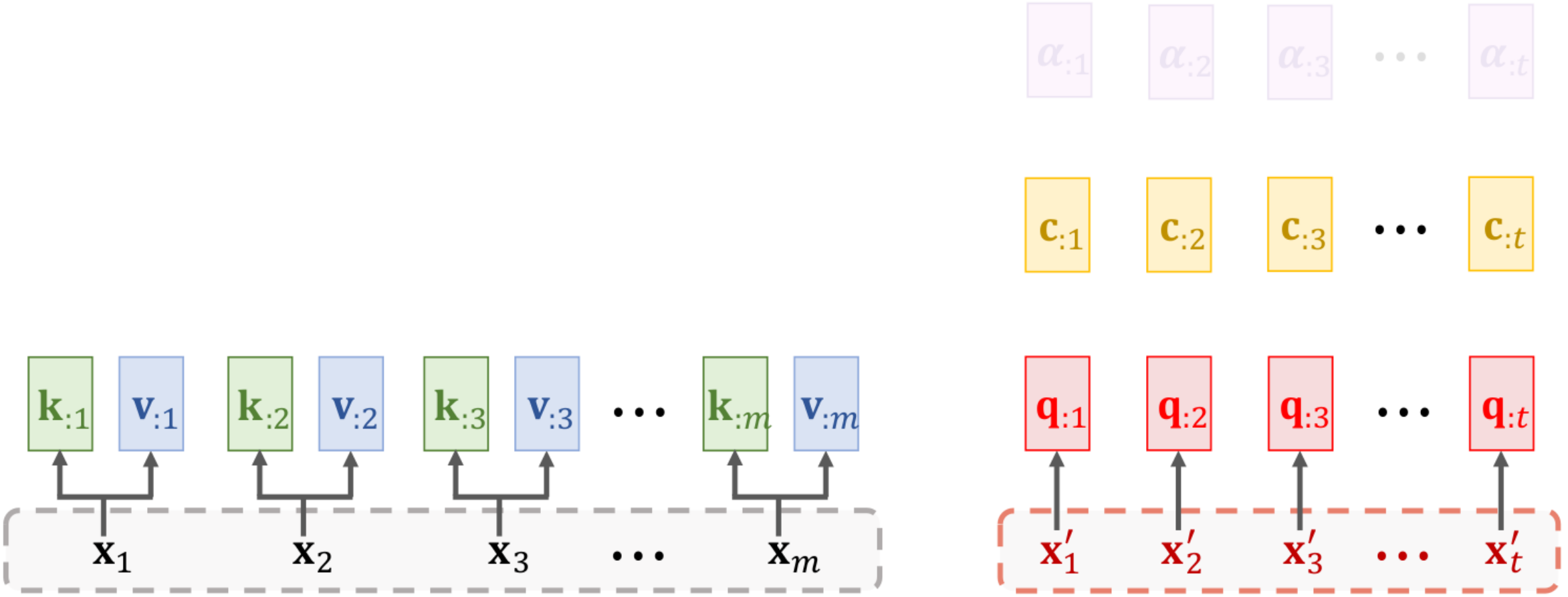
Attention Layer

- Compute weights: $\boldsymbol{\alpha}_{:2} = \text{Softmax}(\mathbf{K}^T \mathbf{q}_{:2}) \in \mathbb{R}^m$.



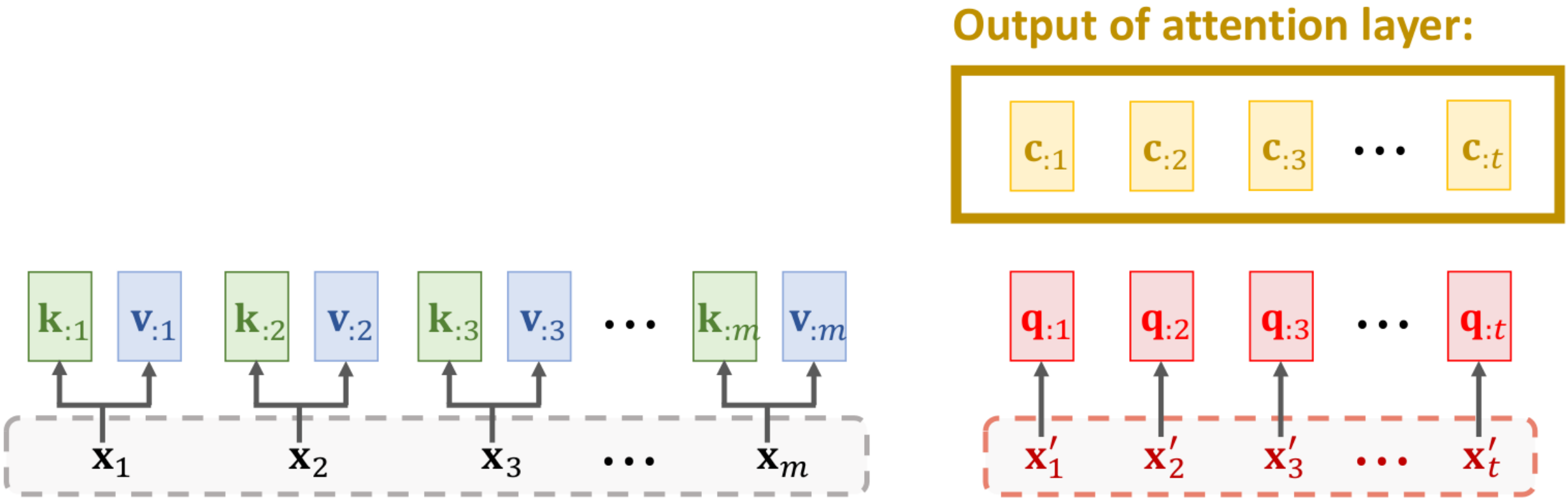
Attention Layer

- Compute context vector: $\mathbf{c}_{:j} = \alpha_{1j}\mathbf{v}_{:1} + \cdots + \alpha_{mj}\mathbf{v}_{:m} = \mathbf{V}\boldsymbol{\alpha}_{:j}$.



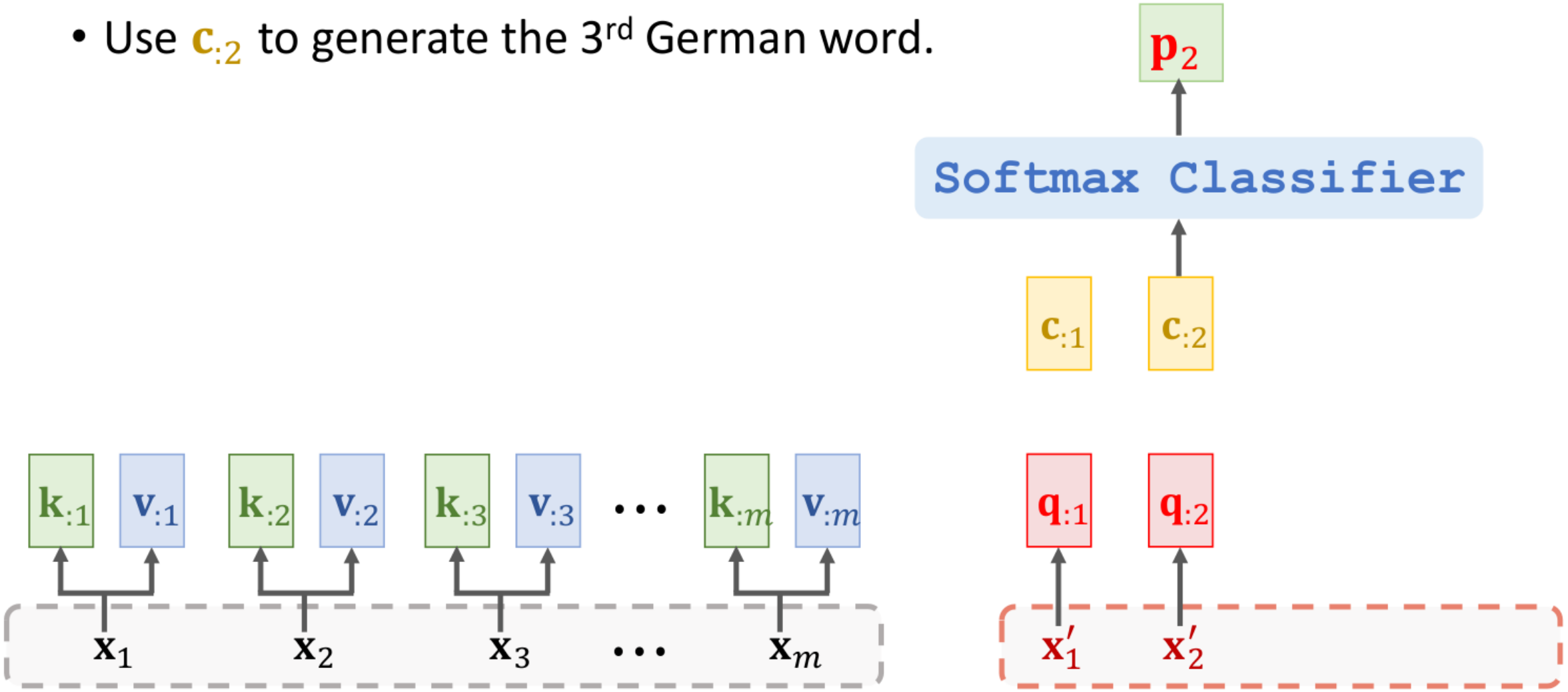
Attention Layer

- Output of attention layer: $\mathbf{C} = [\mathbf{c}_{:1}, \mathbf{c}_{:2}, \mathbf{c}_{:3}, \cdots, \mathbf{c}_{:t}]$.
- Here, $\mathbf{c}_{:j} = \mathbf{V} \cdot \mathrm{Softmax}(\mathbf{K}^T \mathbf{q}_{:j})$.
- Thus, $\mathbf{c}_{:j}$ is a function of $\mathbf{x}'_j$ and $[\mathbf{x}_1, \cdots, \mathbf{x}_m]$.



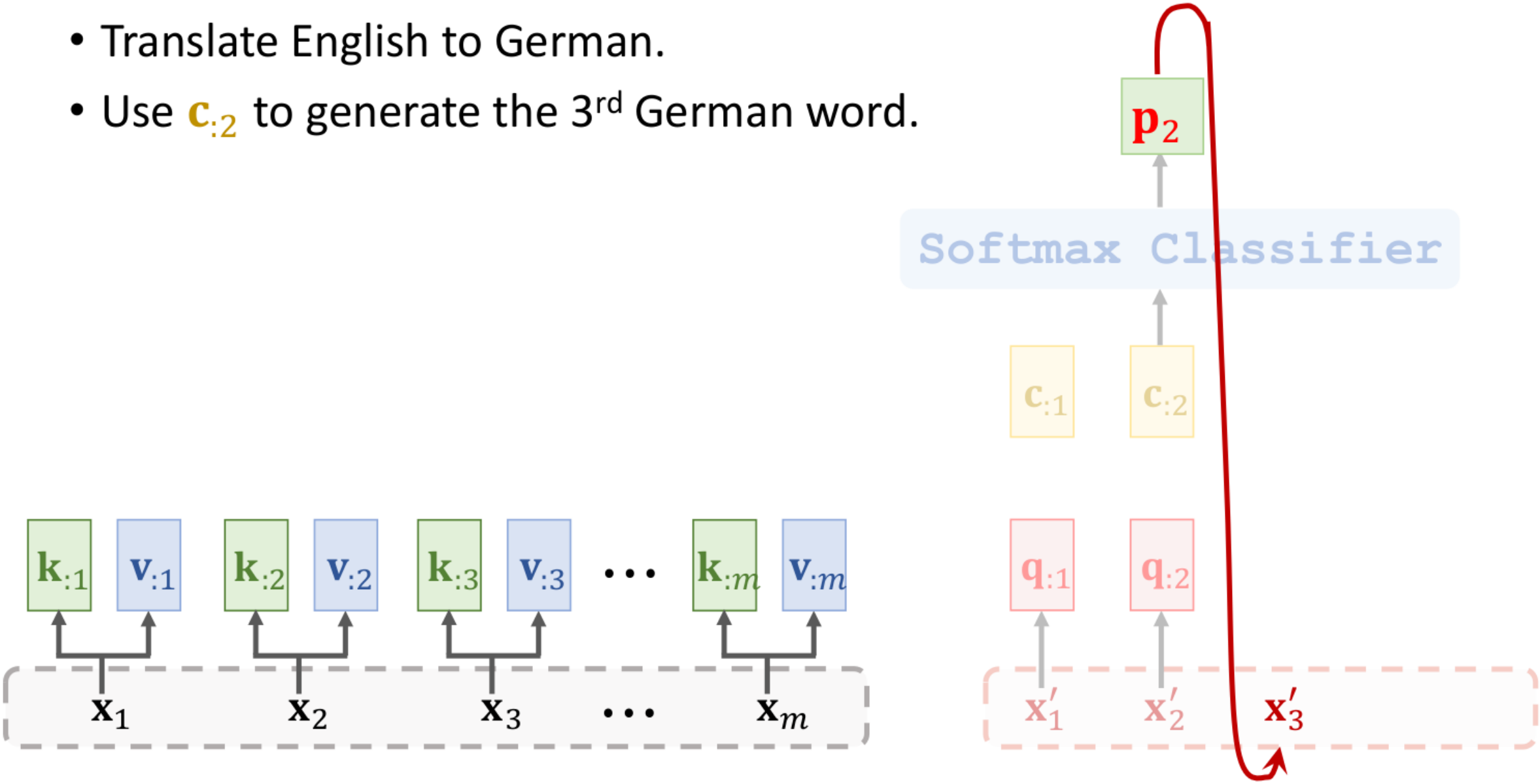
Attention Layer for Machine Translation

- Translate English to German.
- Use $\mathbf{c}_{:2}$ to generate the 3rd German word.



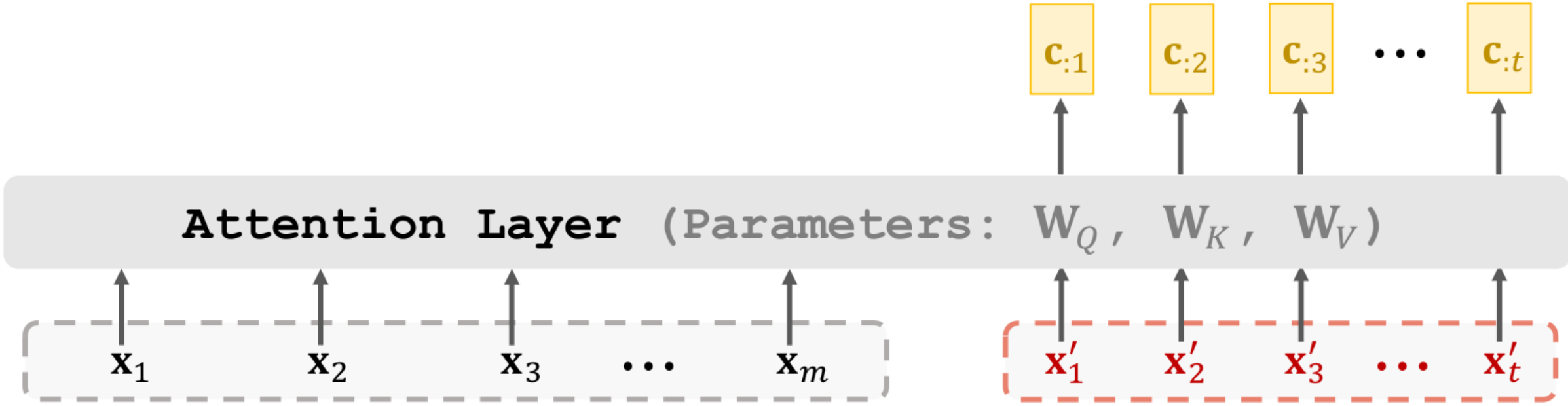
Attention Layer for Machine Translation

- Translate English to German.
- Use $\mathbf{c}_{:2}$ to generate the 3rd German word.



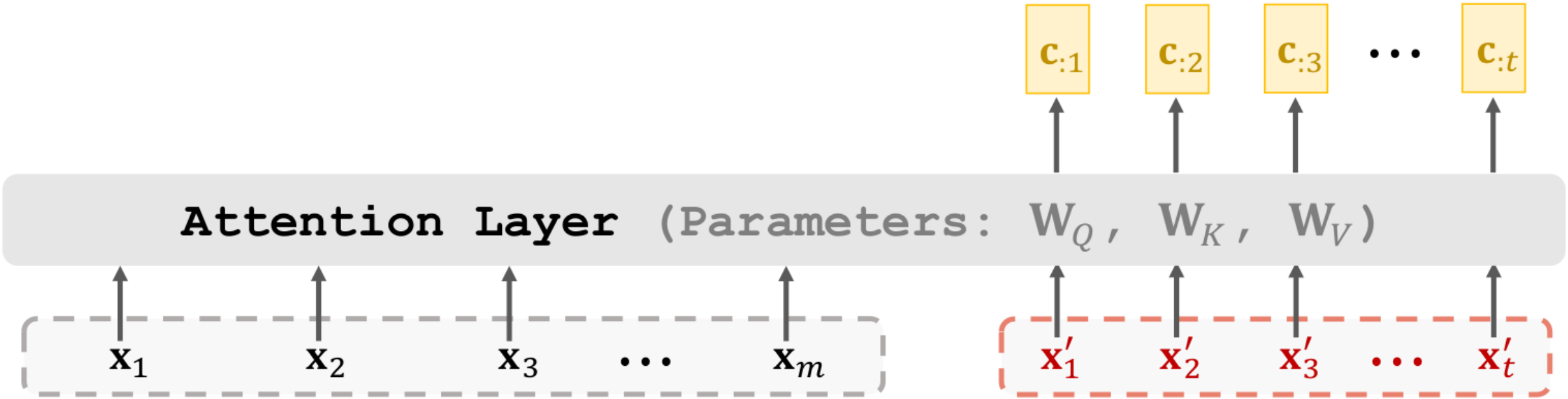
Attention Layer

- Attention layer: $\mathbf{C} = \mathrm{Attn}(\mathbf{X}, \mathbf{X}')$.
  - Encoder's inputs: $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m]$.
  - Decoder's inputs: $\mathbf{X}' = [\mathbf{x}'_1, \mathbf{x}'_2, \cdots, \mathbf{x}'_t]$.
  - Parameters: $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$.



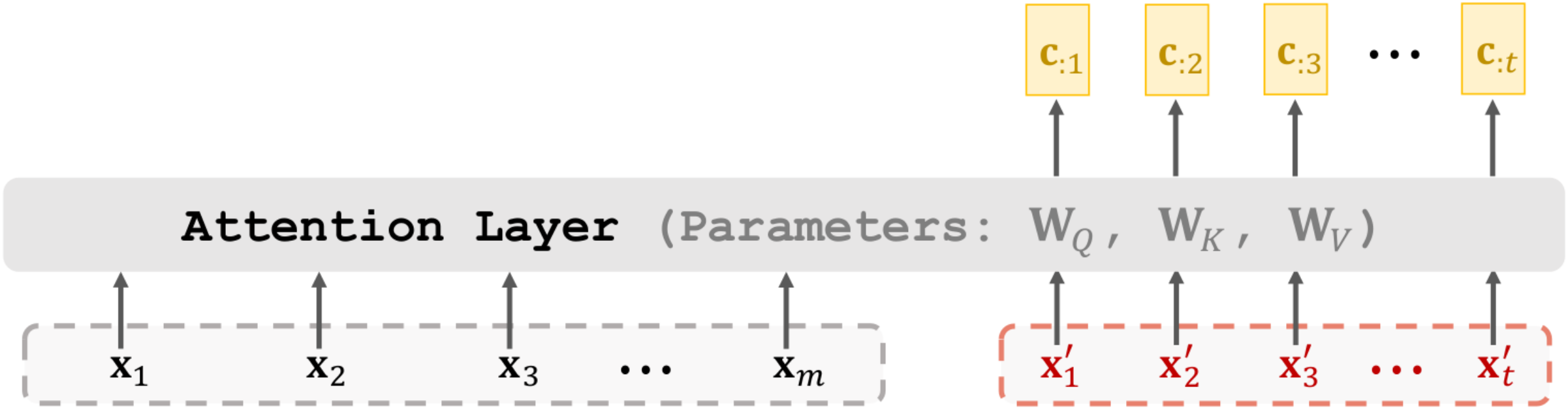
Attention Layer

- Attention layer: $\mathbf{C} = \mathrm{Attn}(\mathbf{X}, \mathbf{X}')$.
  - Encoder's inputs: $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m]$.
  - Decoder's inputs: $\mathbf{X}' = [\mathbf{x}'_1, \mathbf{x}'_2, \cdots, \mathbf{x}'_m]$.
  - Parameters: $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$.



Self-Attention Layer

- Attention layer: $\mathbf{C} = \mathrm{Attn}(\mathbf{X}, \mathbf{X}')$.
  - Encoder's inputs: $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m]$.
  - Decoder's inputs: $\mathbf{X}' = [\mathbf{x}'_1, \mathbf{x}'_2, \cdots, \mathbf{x}'_m]$.
  - Parameters: $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$.

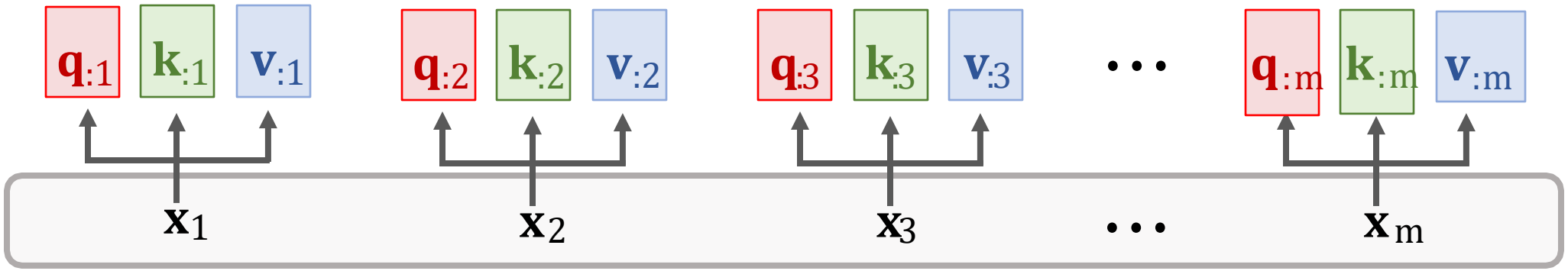


Self-Attention Layer

Inputs:

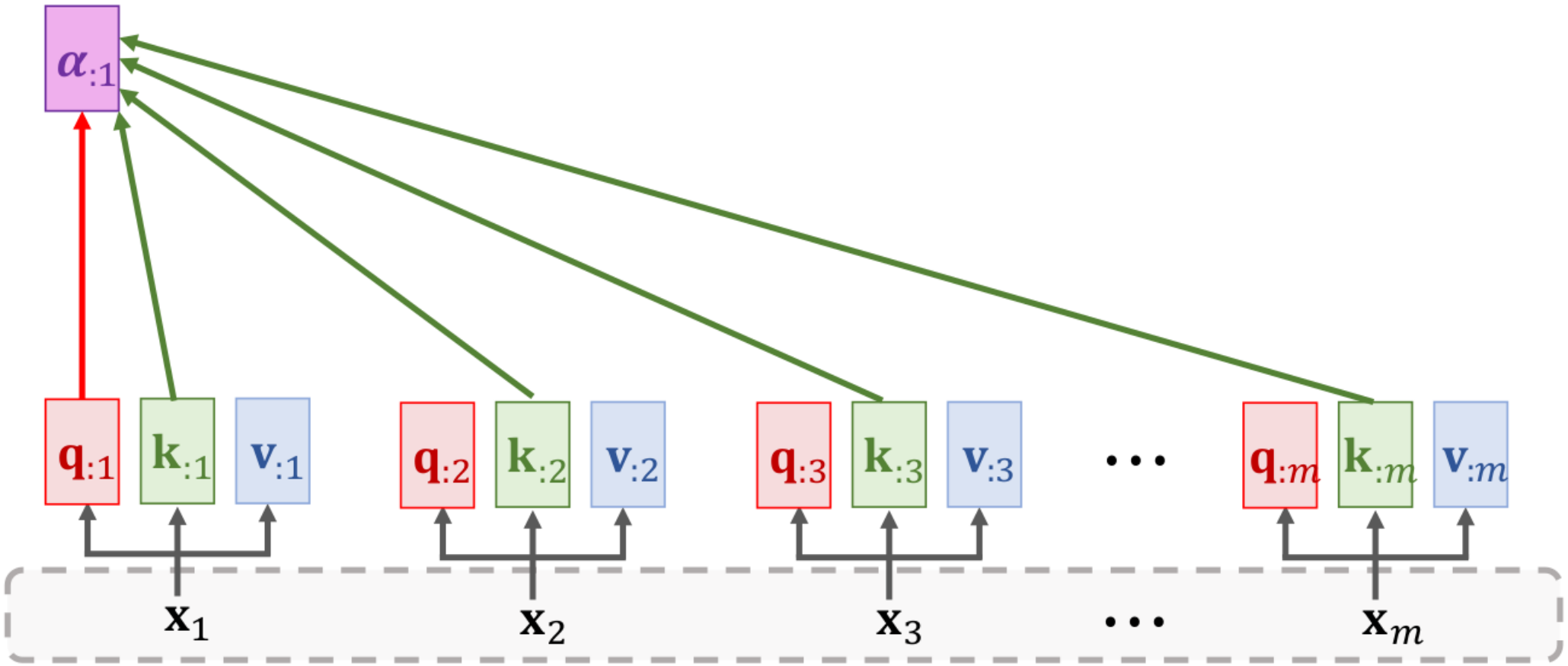
Self-Attention Layer

Query: $\mathbf{q}_{:i} = \mathbf{W}_Q \mathbf{x}_i$, Key: $\mathbf{k}_{:i} = \mathbf{W}_k \mathbf{x}_i$, Value: $\mathbf{v}_{:i} = \mathbf{W}_v \mathbf{x}_i$.



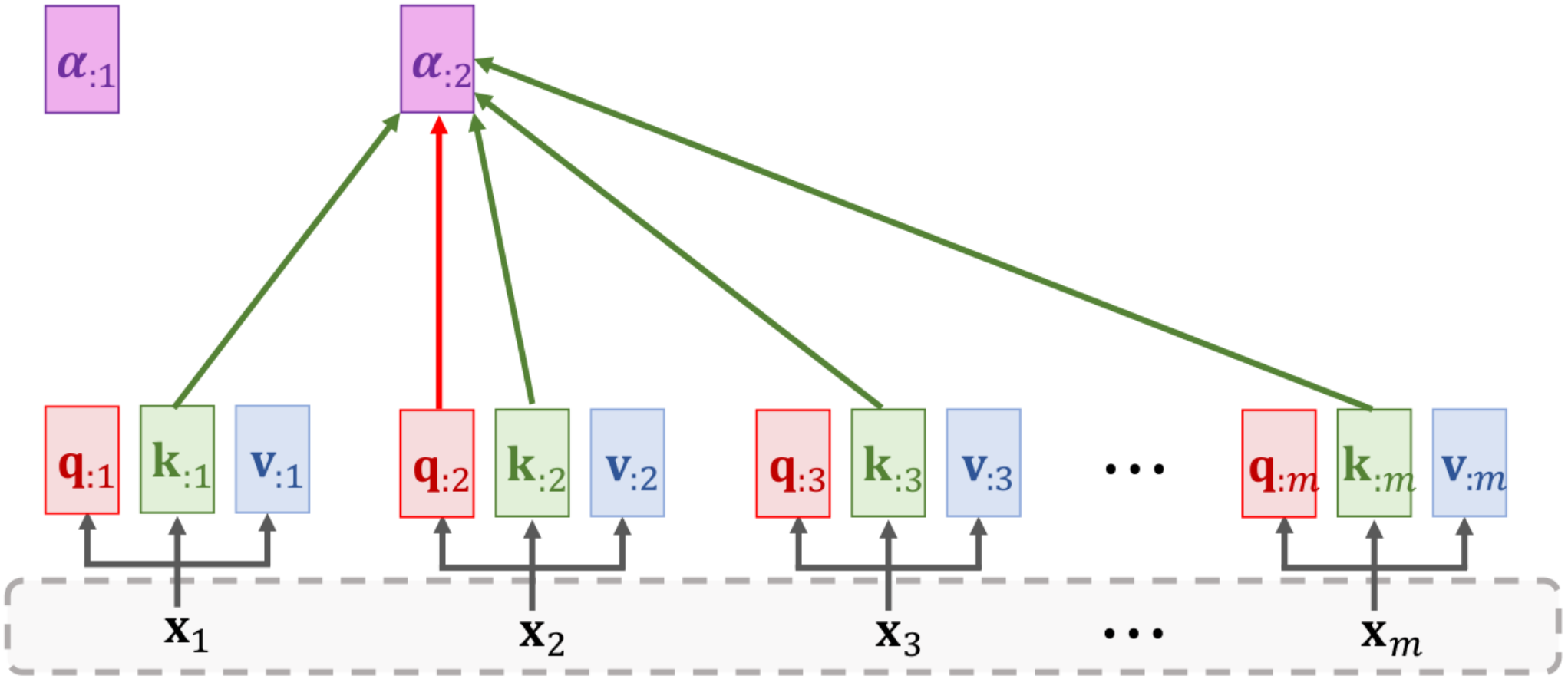
Self-Attention Layer

Weights: $\alpha_{:j} = \text{Softmax}\left(\mathbf{K}^T \mathbf{q}_{:j}\right) \in \mathbb{R}^m.$



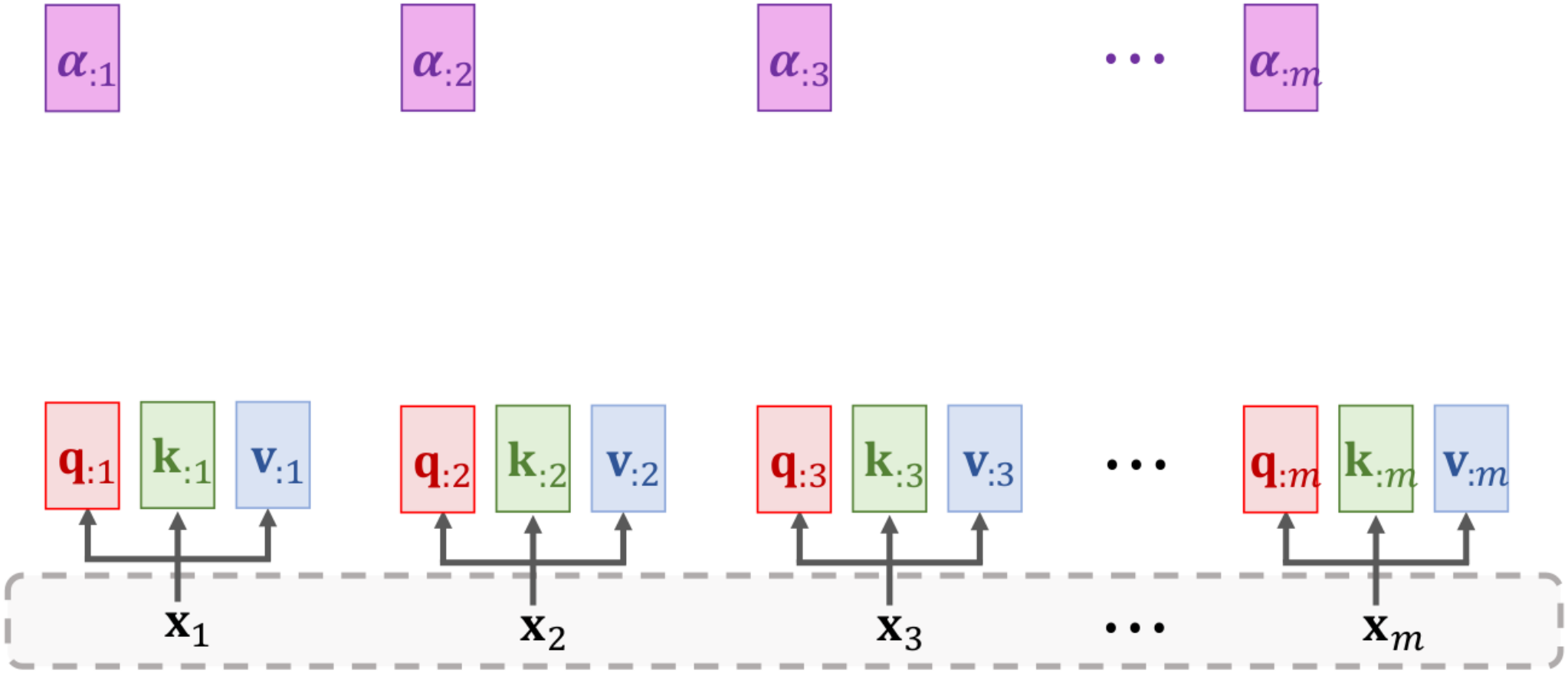
Self-Attention Layer

Weights: $\alpha_{:j} = \text{Softmax}\left(\mathbf{K}^T \mathbf{q}_{:j}\right) \in \mathbb{R}^m.$



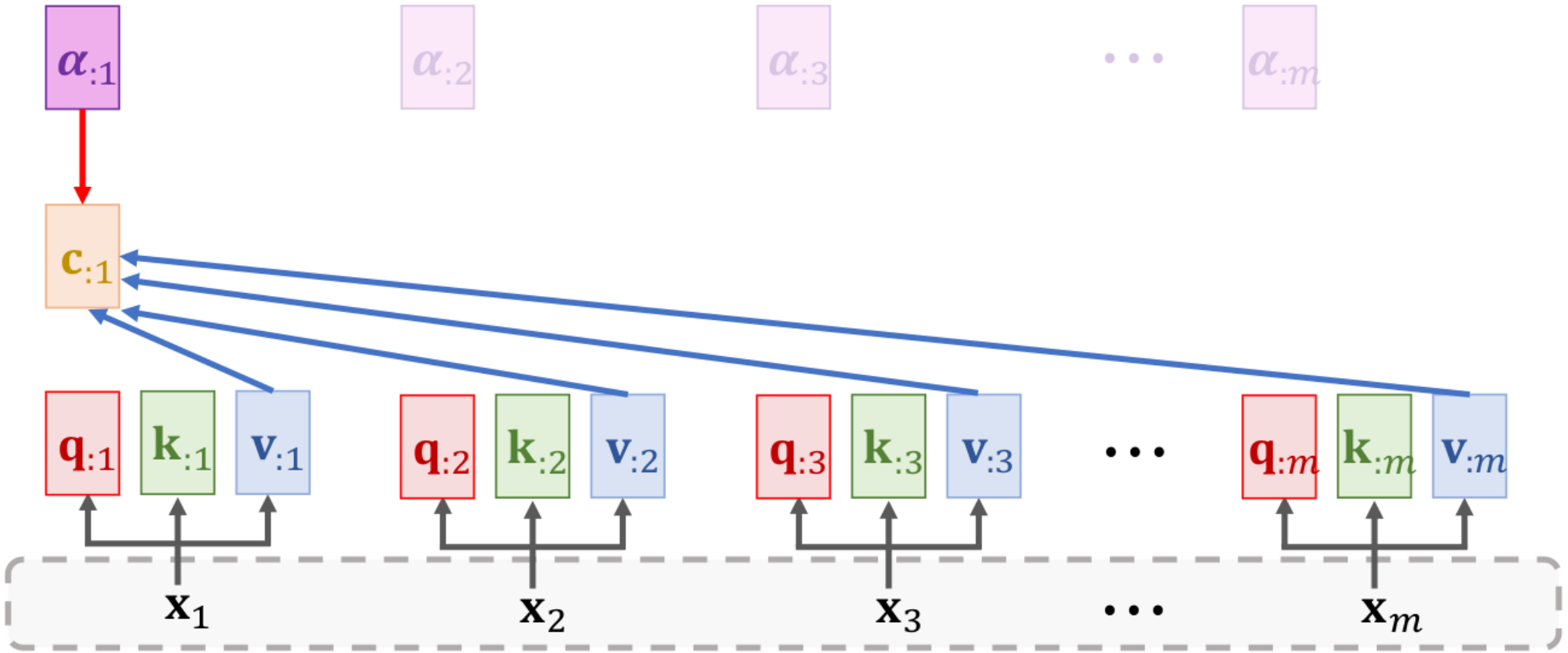
Self-Attention Layer

Weights: $\alpha_{:j} = \text{Softmax}\left(\mathbf{K}^T \mathbf{q}_{:j}\right) \in \mathbb{R}^m.$



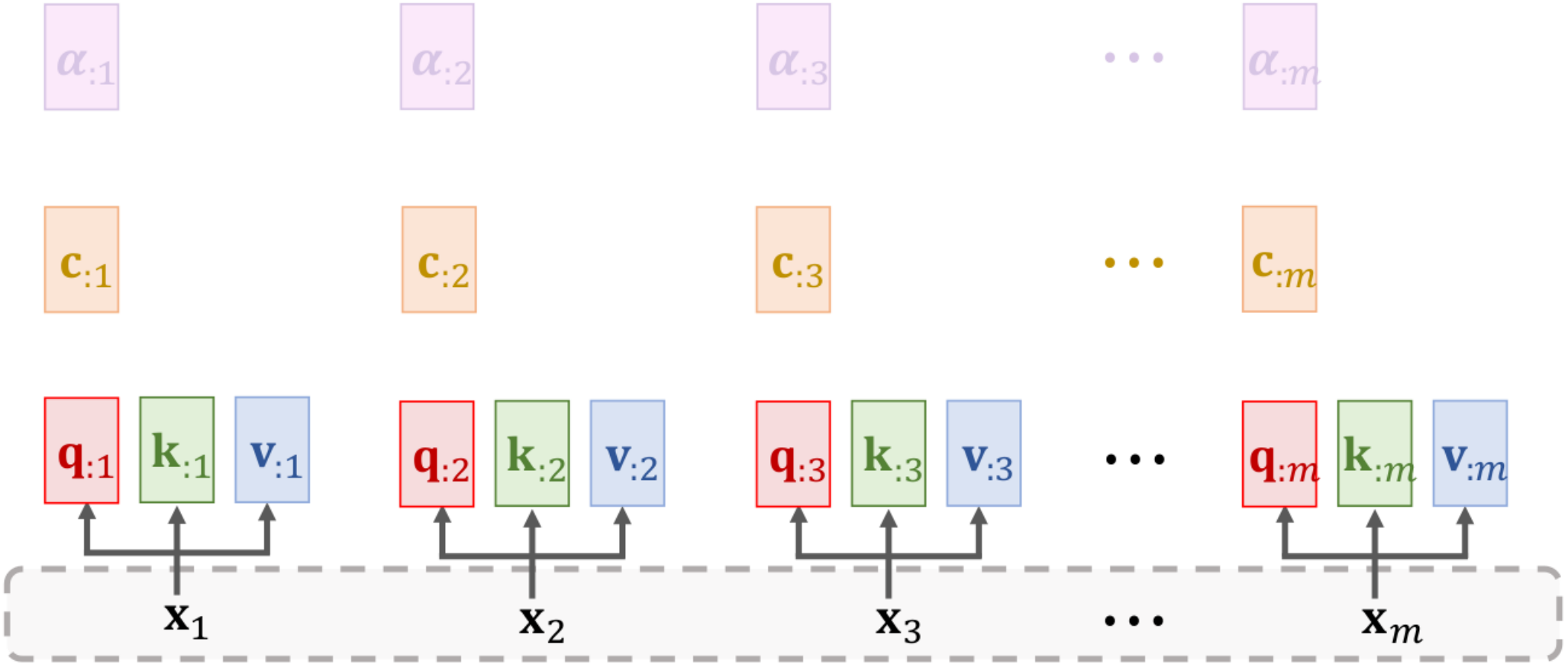
Self-Attention Layer

Context vector: $\mathbf{c}_{:1} = \alpha_{11}\mathbf{v}_{:1} + \cdots + \alpha_{m1}\mathbf{v}_{:m} = \mathbf{V}\boldsymbol{\alpha}_{:1}.$



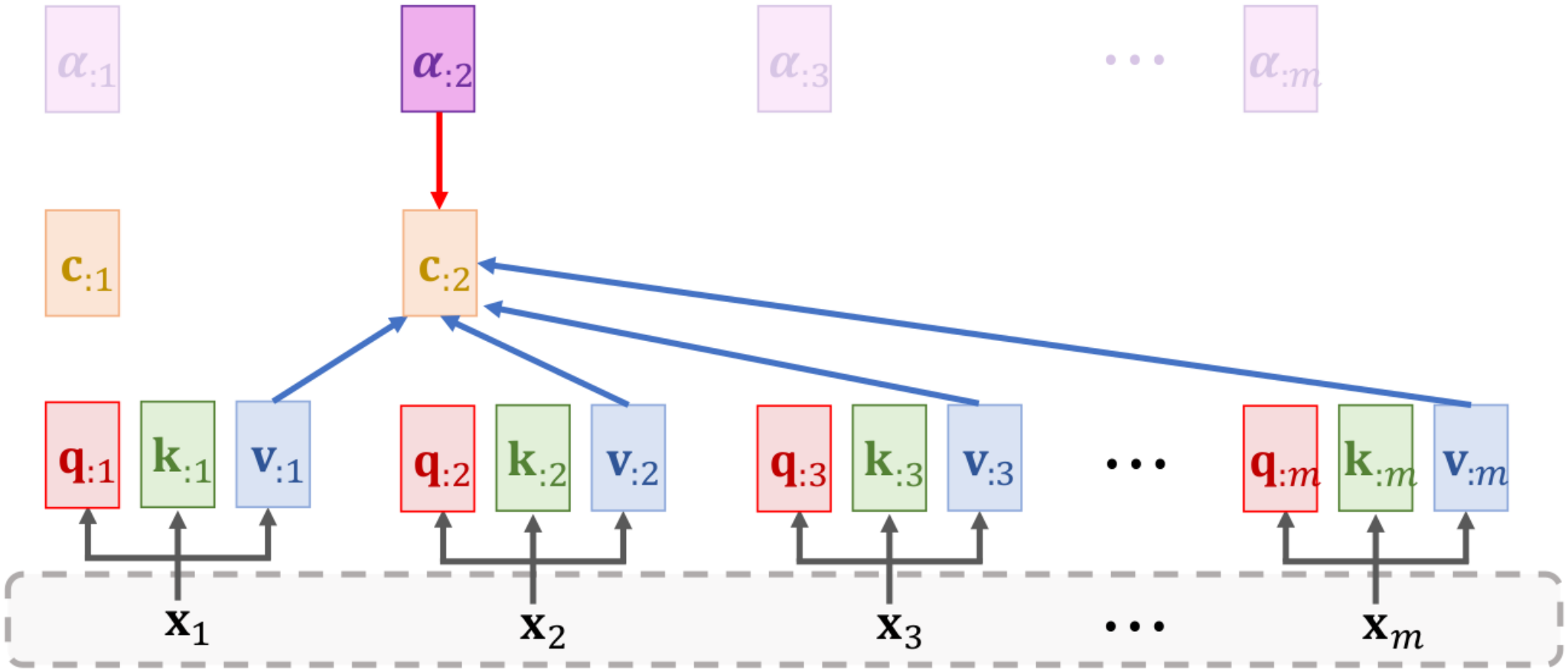
Self-Attention Layer

Context vector: $\mathbf{c}_{:j} = \alpha_{1j}\mathbf{v}_{:1} + \cdots + \alpha_{mj}\mathbf{v}_{:m} = \mathbf{V}\boldsymbol{\alpha}_{:j}.$



Self-Attention Layer

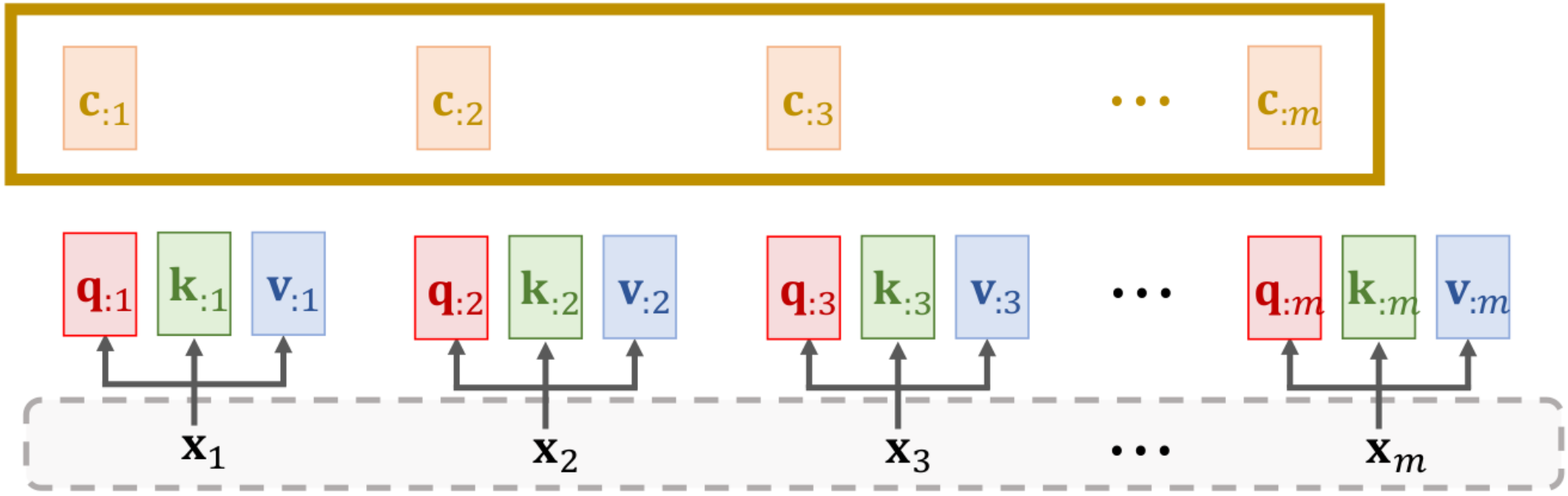
Context vector: $\mathbf{c}_{:2} = \alpha_{12}\mathbf{v}_{:1} + \cdots + \alpha_{m2}\mathbf{v}_{:m} = \mathbf{V}\boldsymbol{\alpha}_{:2}.$



Self-Attention Layer

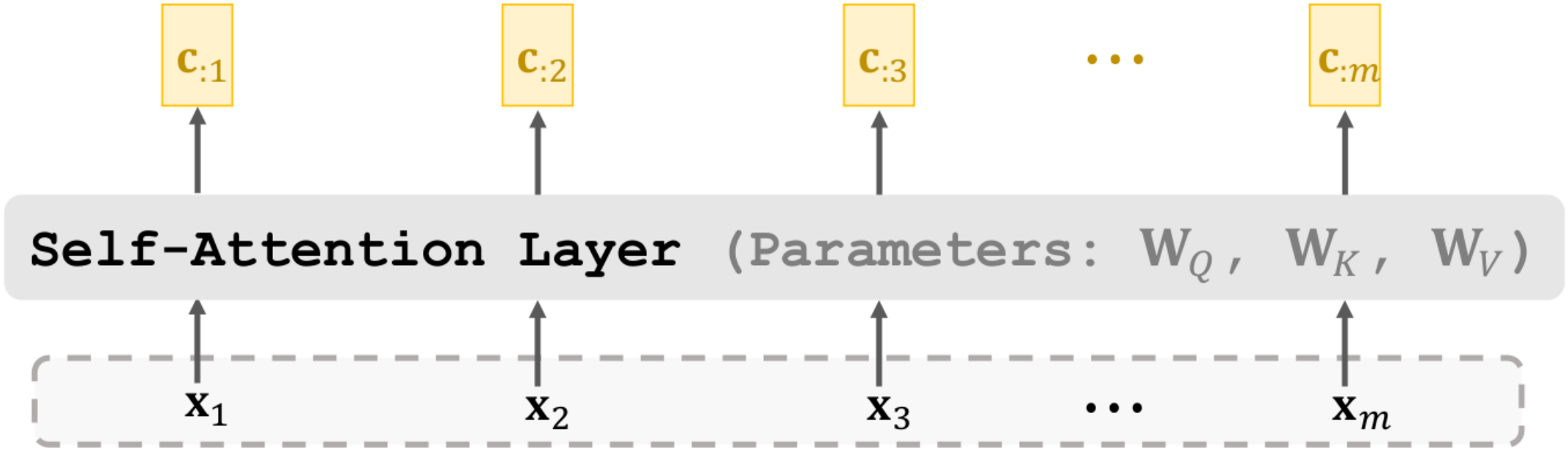
- Ovde, $\mathbf{c}_{:j} = \mathbf{V} \cdot \text{Softmax}\left(\mathbf{K}^T \mathbf{q}_{:j}\right)$.
- Tako, $\mathbf{c}_{:j}$ je funkcija od svih m vektora $\mathbf{x}_1, \cdots, \mathbf{x}_m$.

Output of self-attention layer:

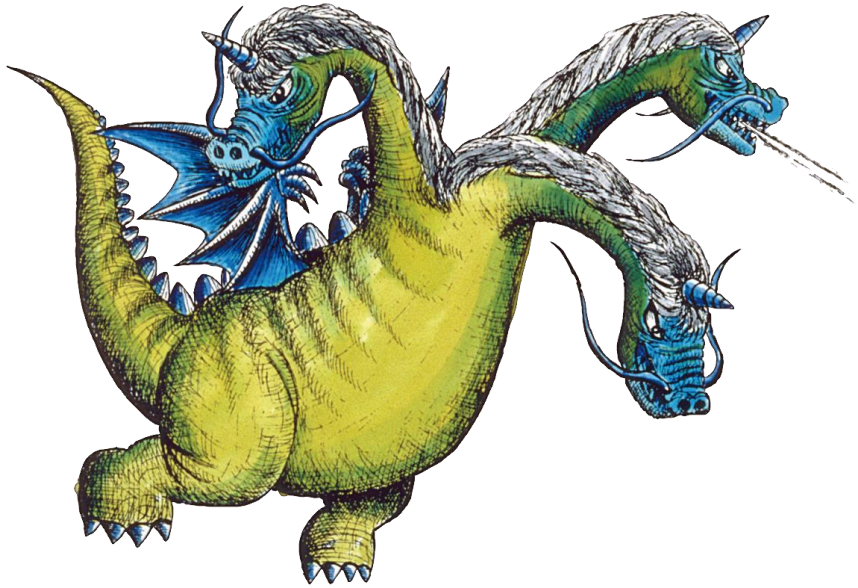


Self-Attention Layer

- Self-attention layer: $\mathbf{C} = \mathrm{Attn}(\mathbf{X}, \mathbf{X})$.
  - Inputs: $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m]$.
  - Parameters: $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$.

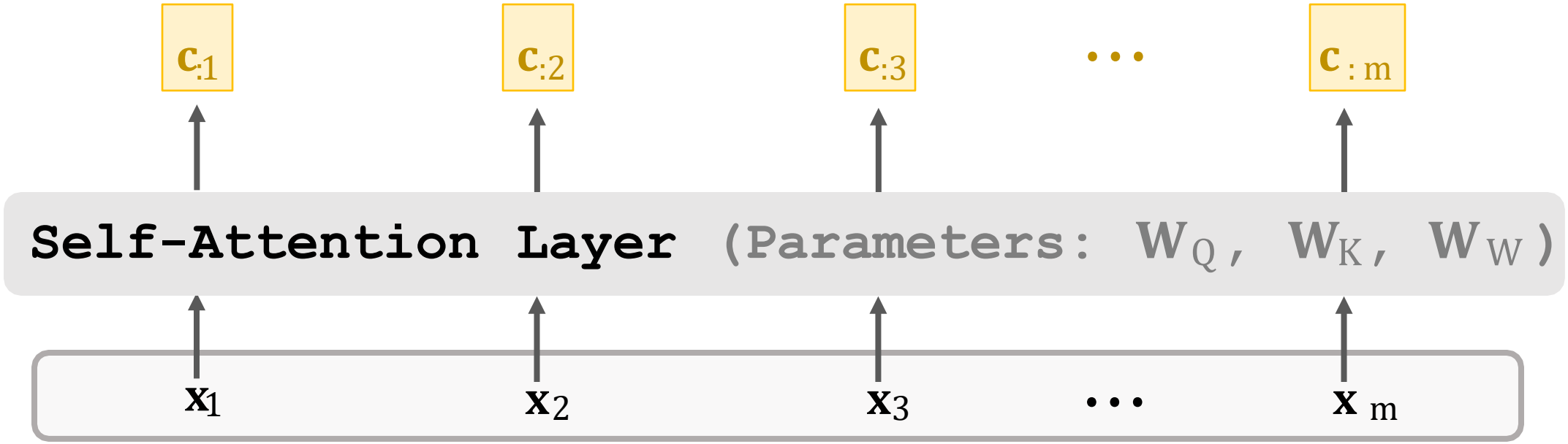


Vise-Glava Paznja



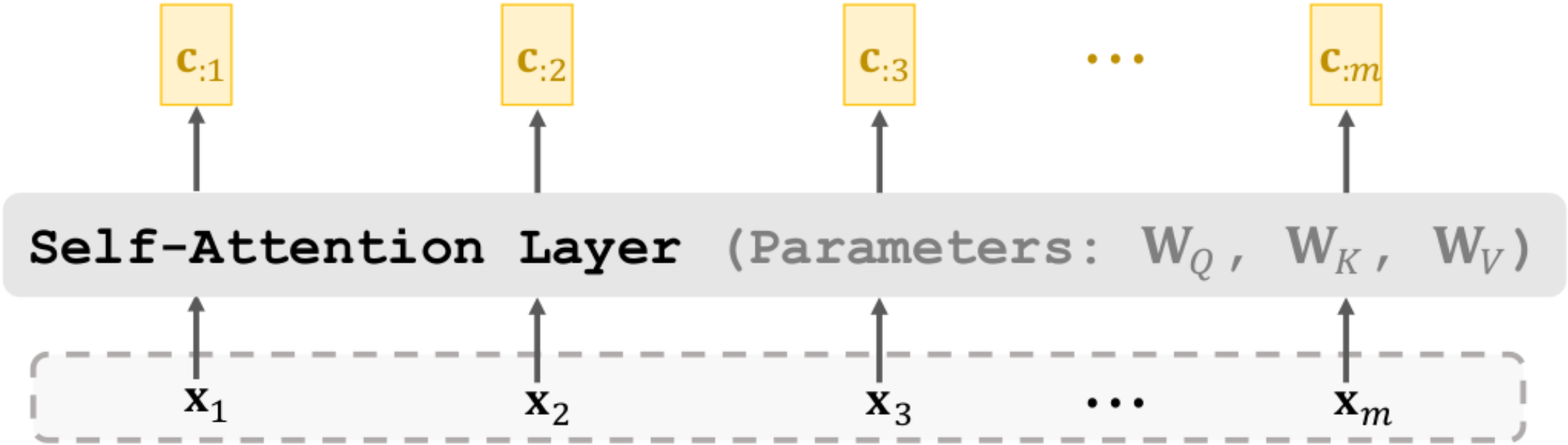
Single-Head Self-Attention

- Self-attention layer: $\mathbf{C} = \mathrm{Attn}(\mathbf{X}, \mathbf{X})$.
- Ovo se zove "single-head self-attention".



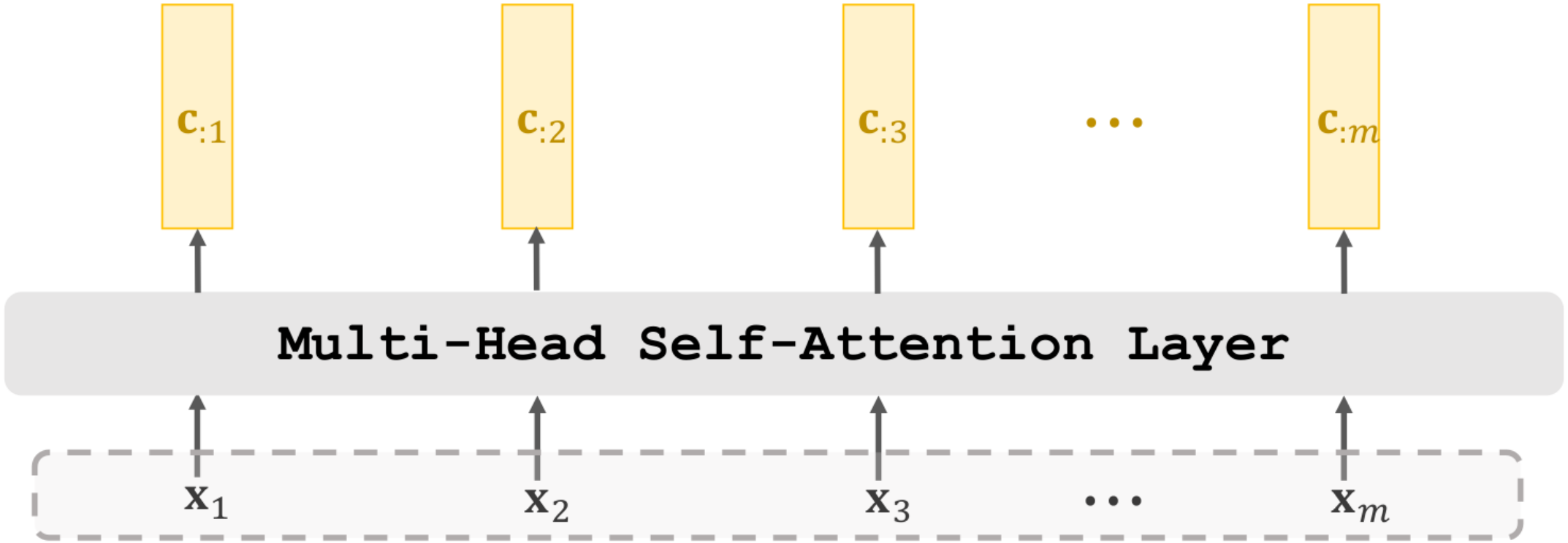
Multi-Head Self-Attention

- Koristeci l single-head self-attentions (koji ne dele parametre)
  - Single-head self-attention ima 3 parametne matricee: $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_W$.
  - Ukupno $3l$ parametne matricee.



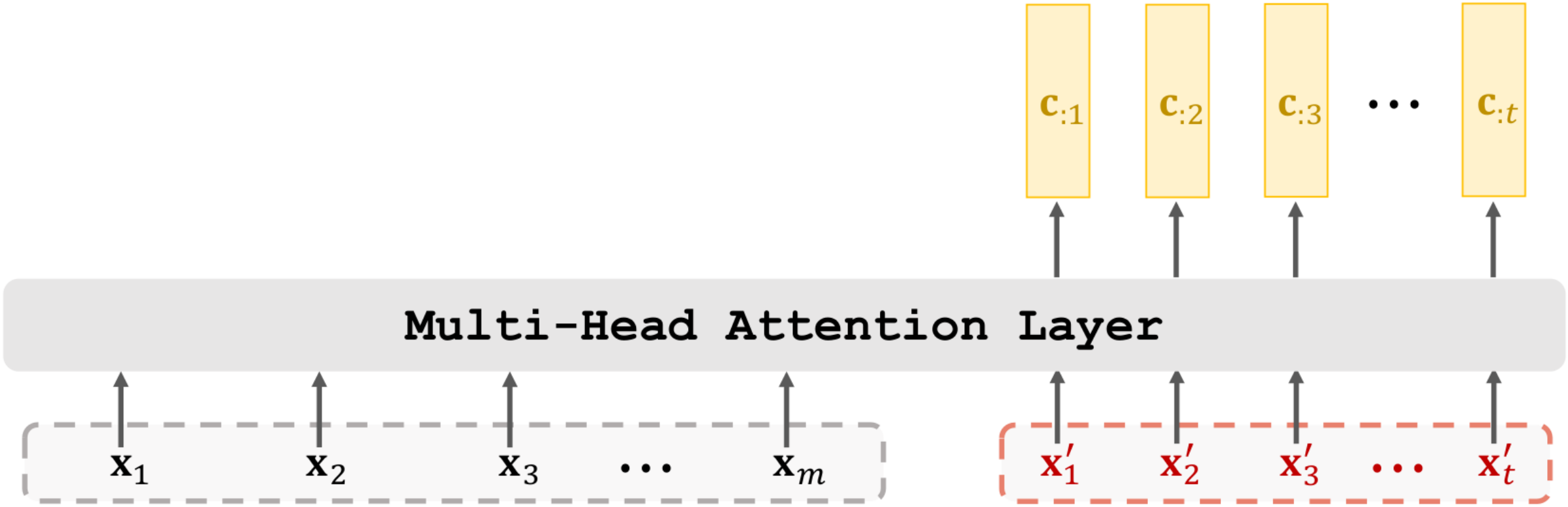
Multi-Head Self-Attention

- Koristeci l single-head self-attentions (koji ne dele parametre.)
- Spaja se izlaz od single-head self-attentions.
  - Pretpostavimo da su single-head self-attentions' izlazi $d \times m$ matrice.
  - Multi-head oblik izlaza: $(ld) \times m$.



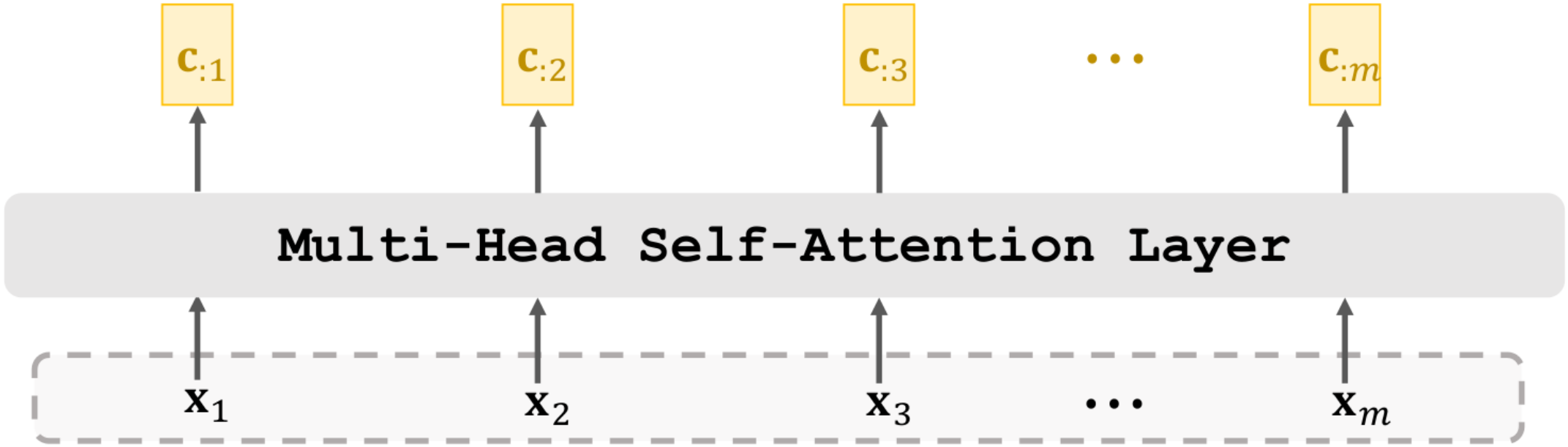
Multi-Head Attention

- Koristeci l single-head attentions (koji ne dele parametre.)
- Spajaju se single-head attentions izlazi.

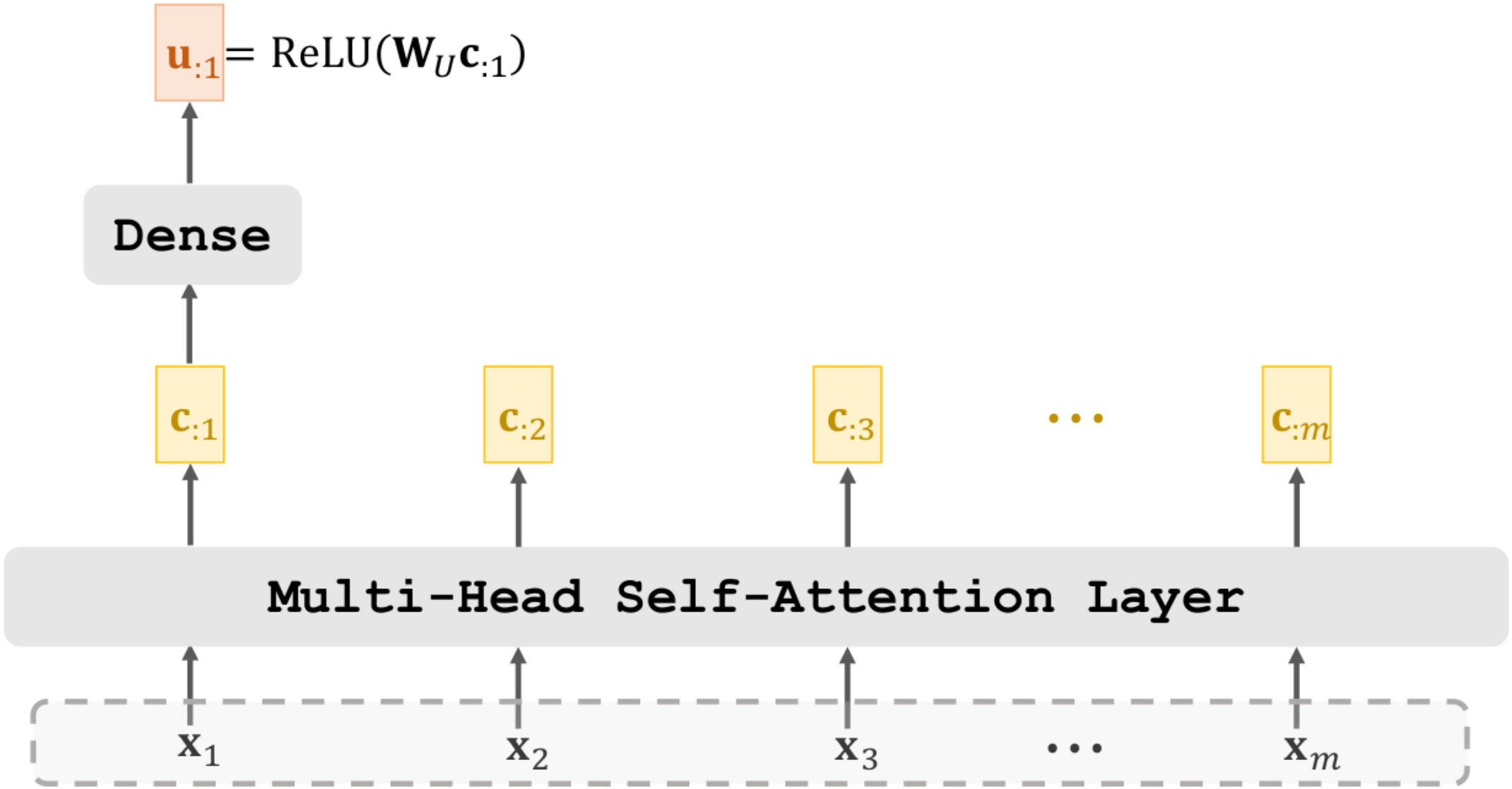


Stacked Self-Attention slojevi

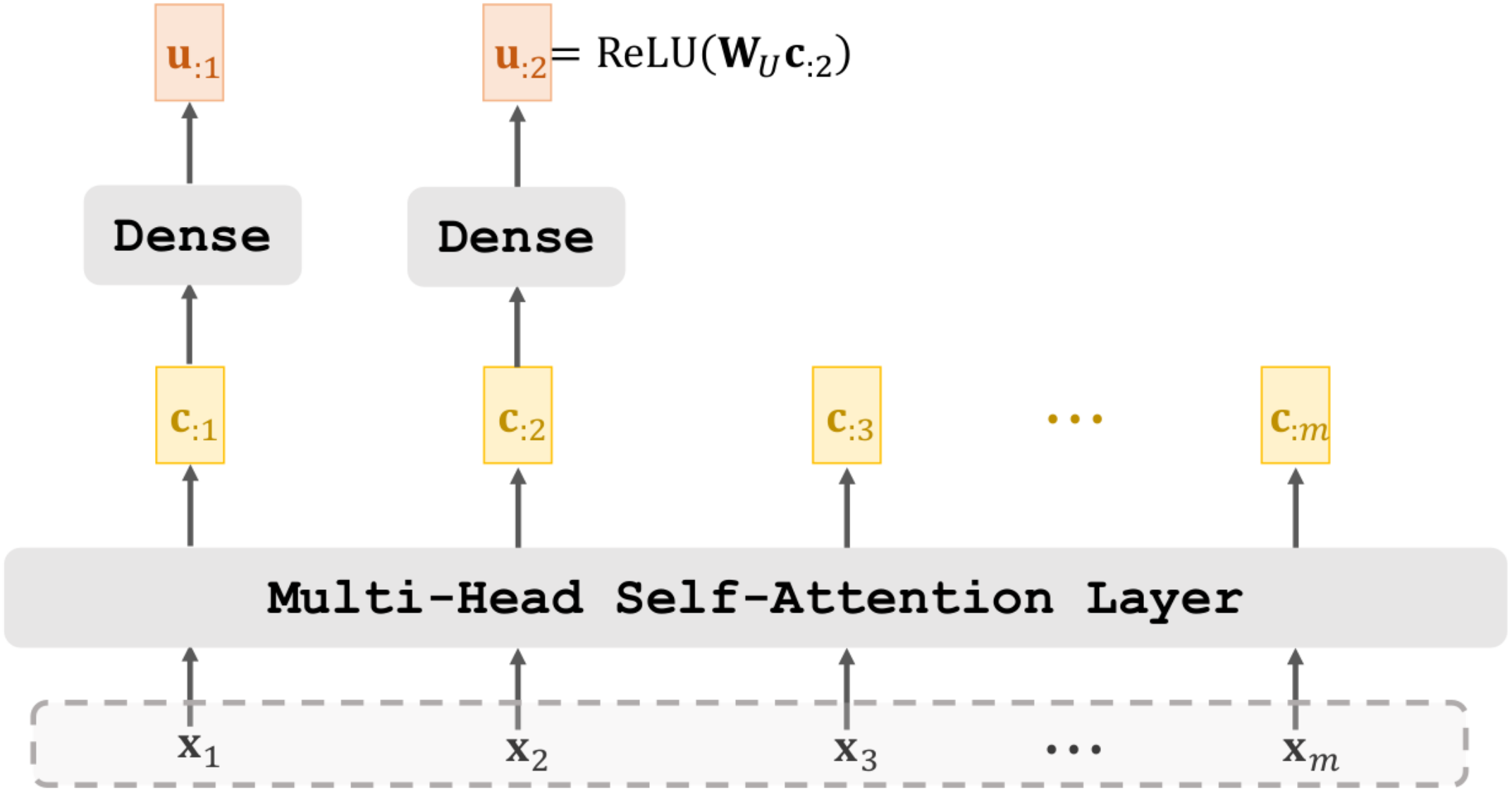
Self-Attention Layer



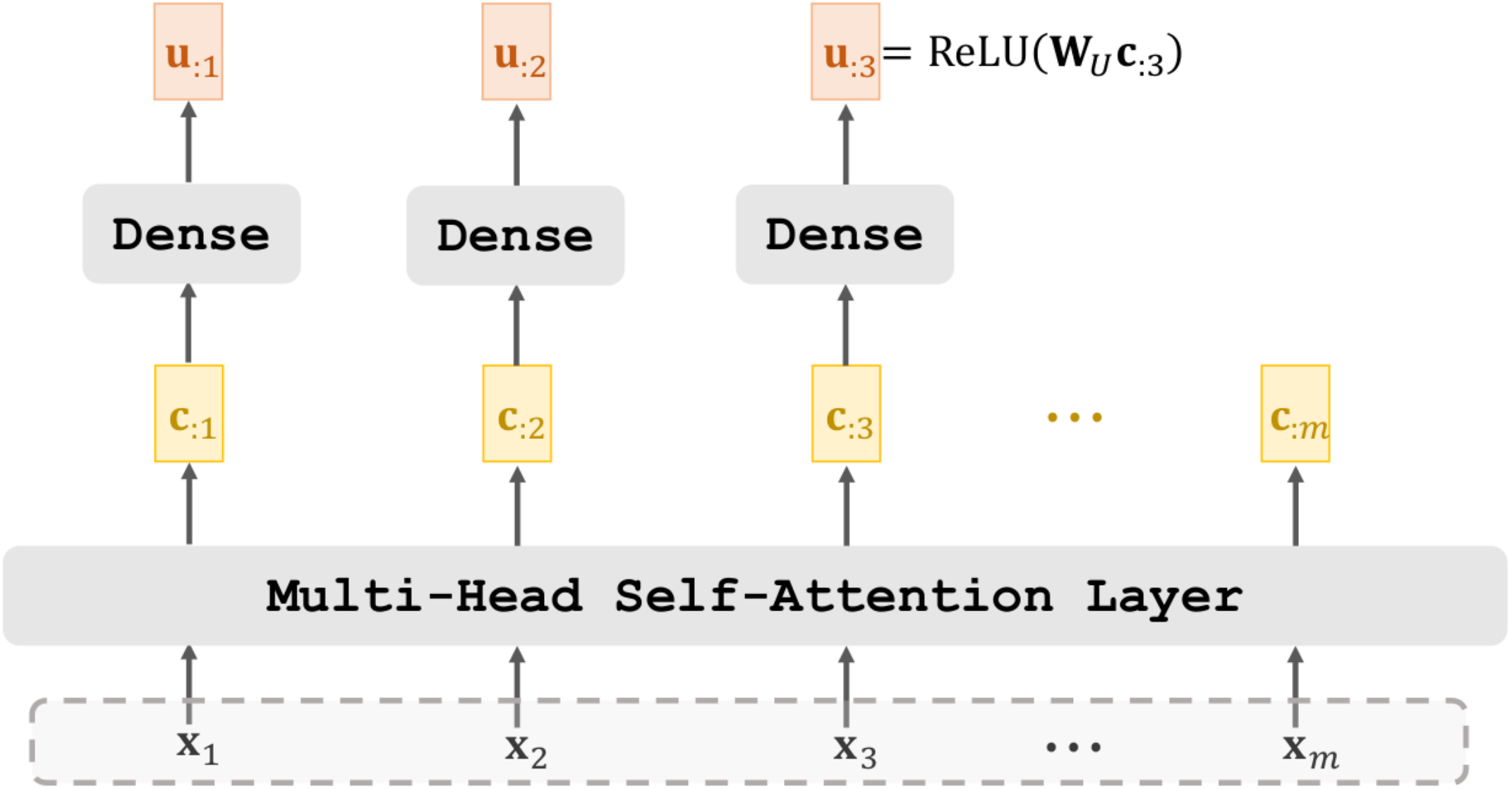
Self-Attention Layer + Dense Layer



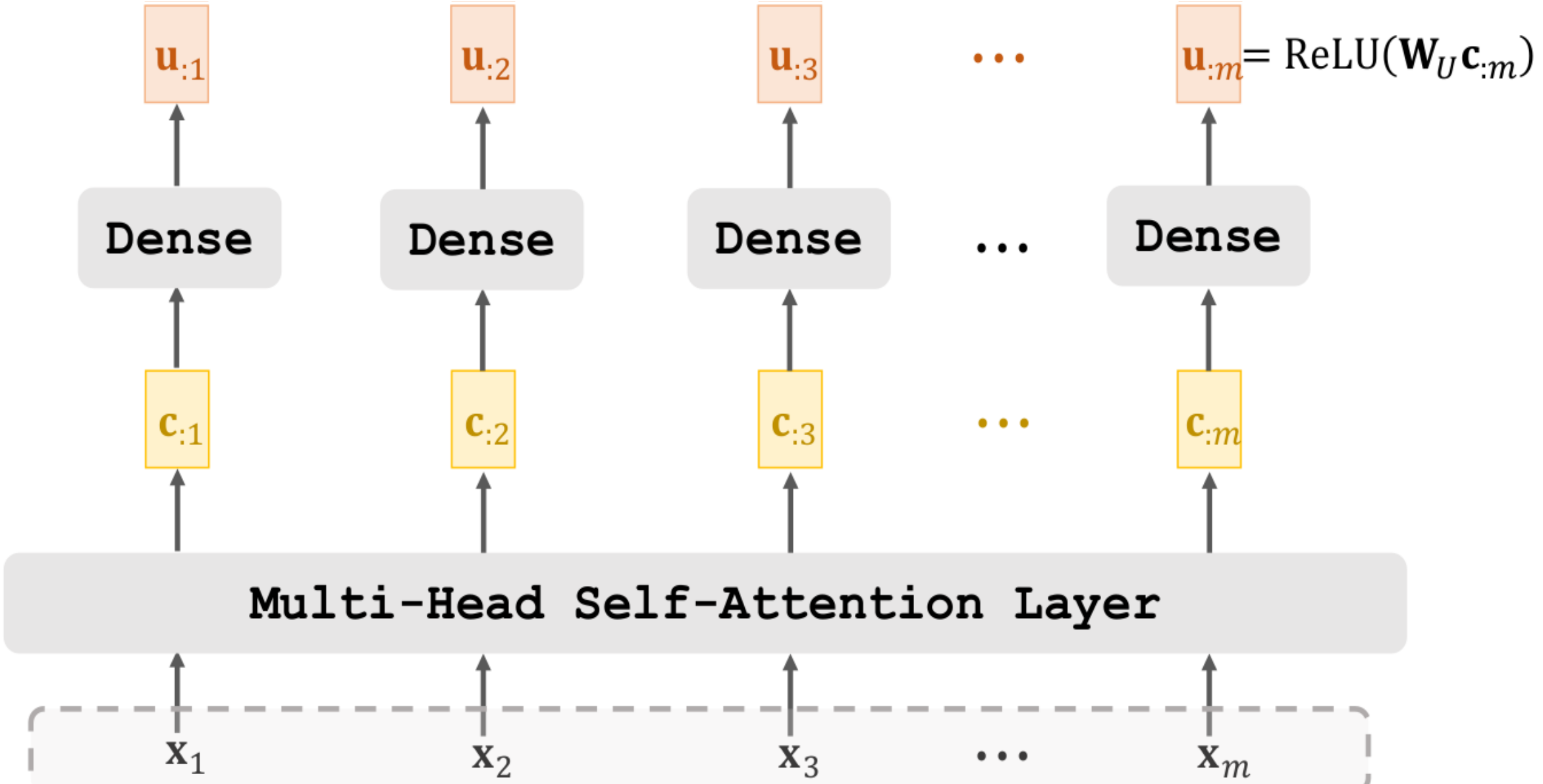
Self-Attention Layer + Dense Layer



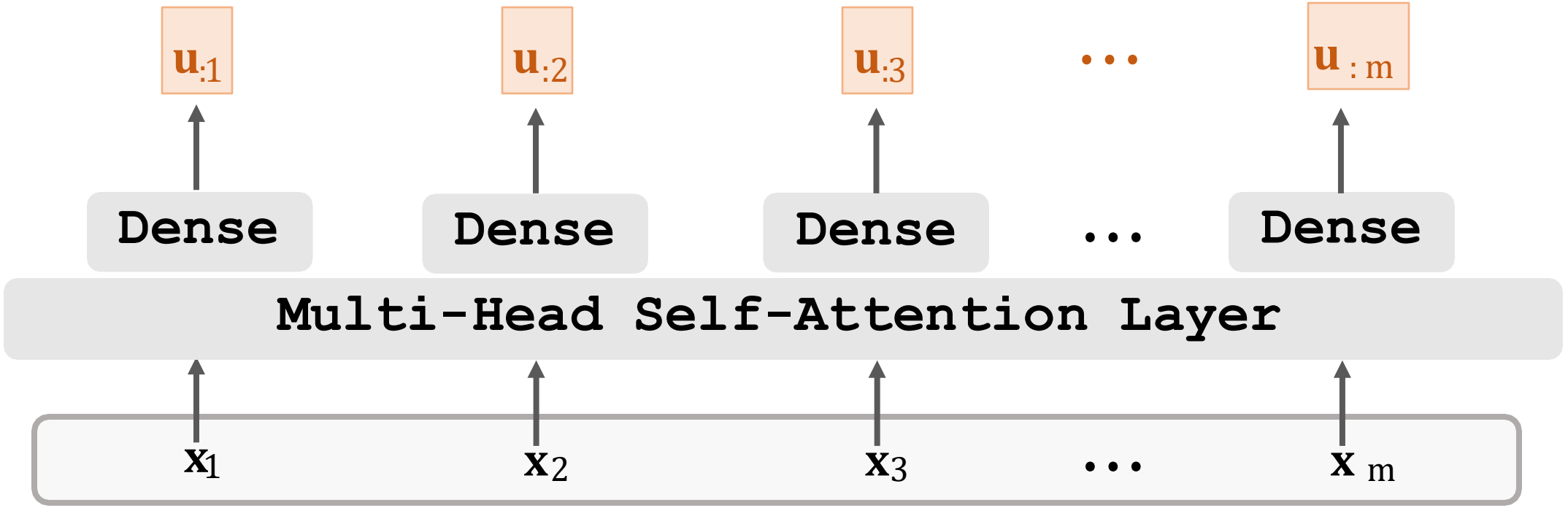
Self-Attention Layer + Dense Layer



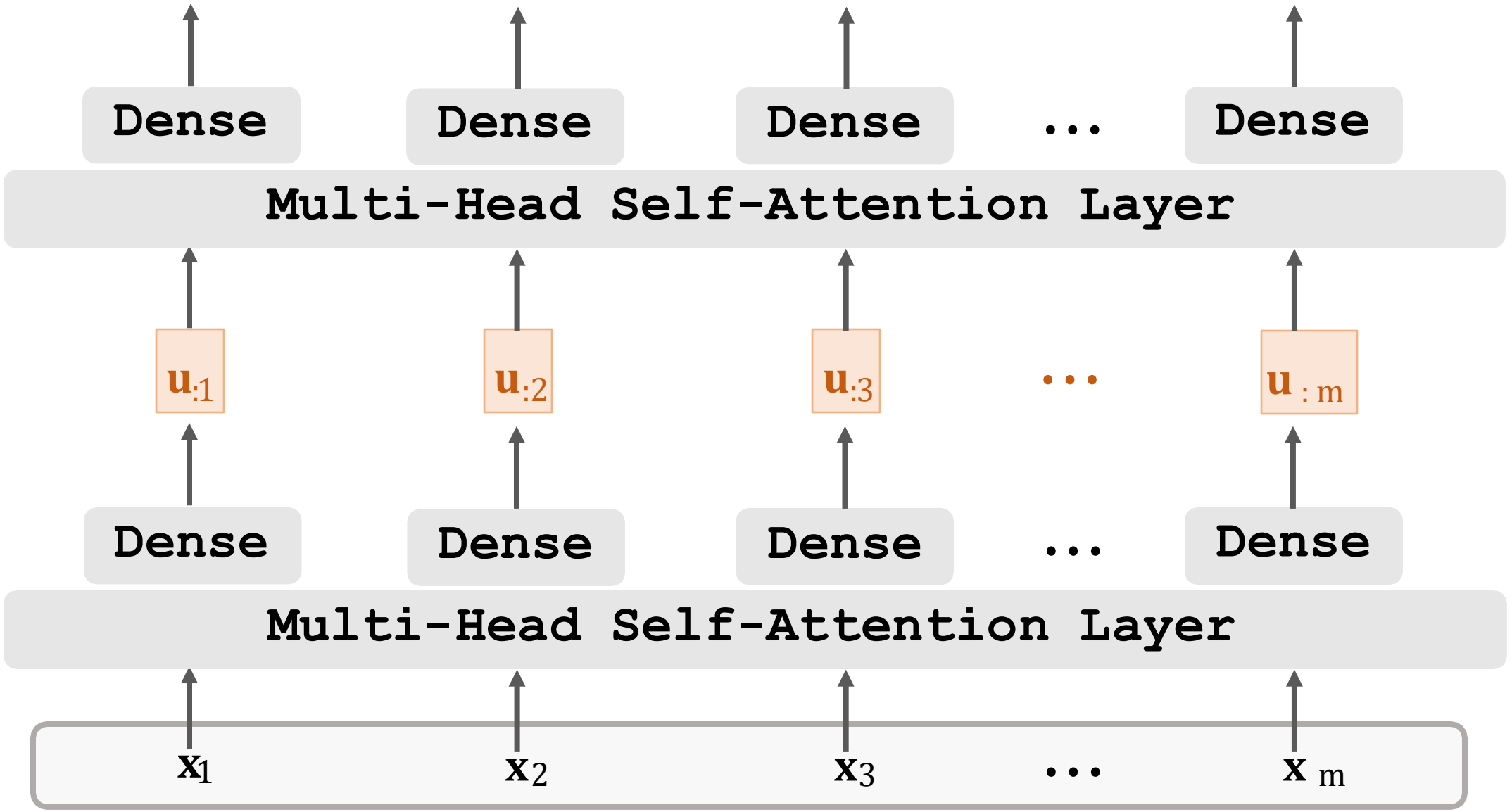
Self-Attention Layer + Dense Layer



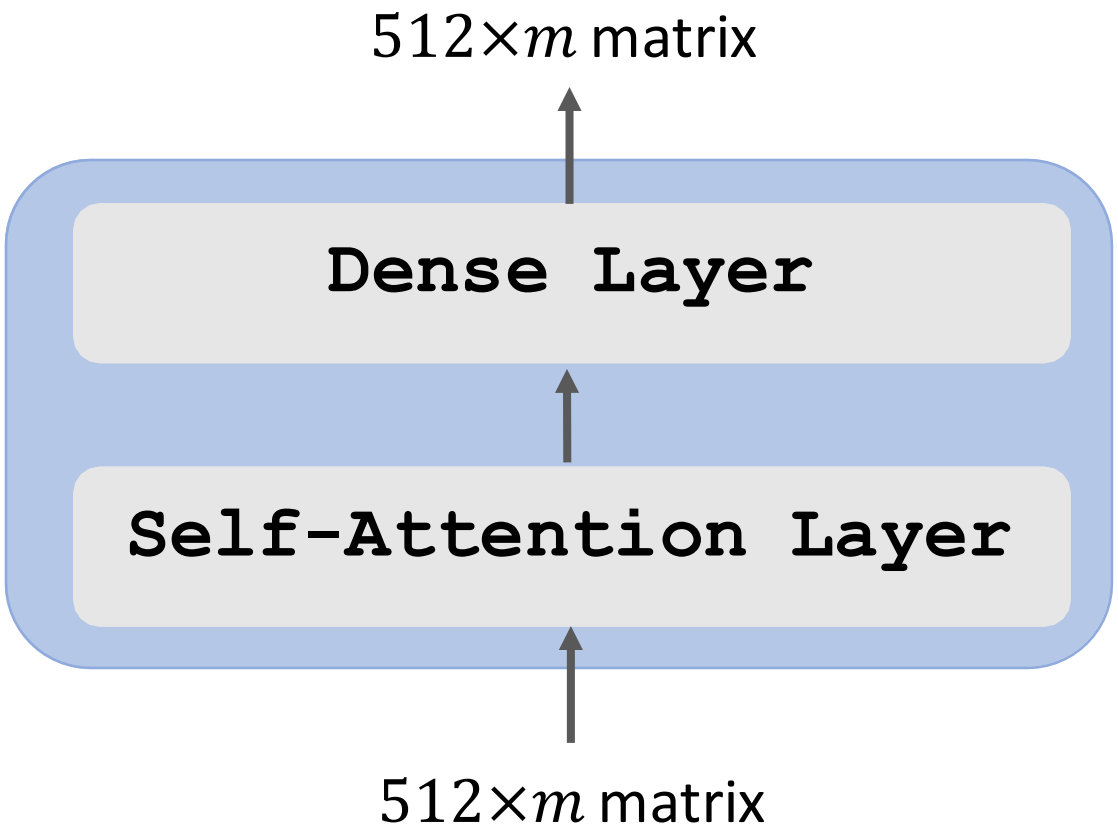
Stacked Self-Attention Layers



Stacked Self-Attention Layers

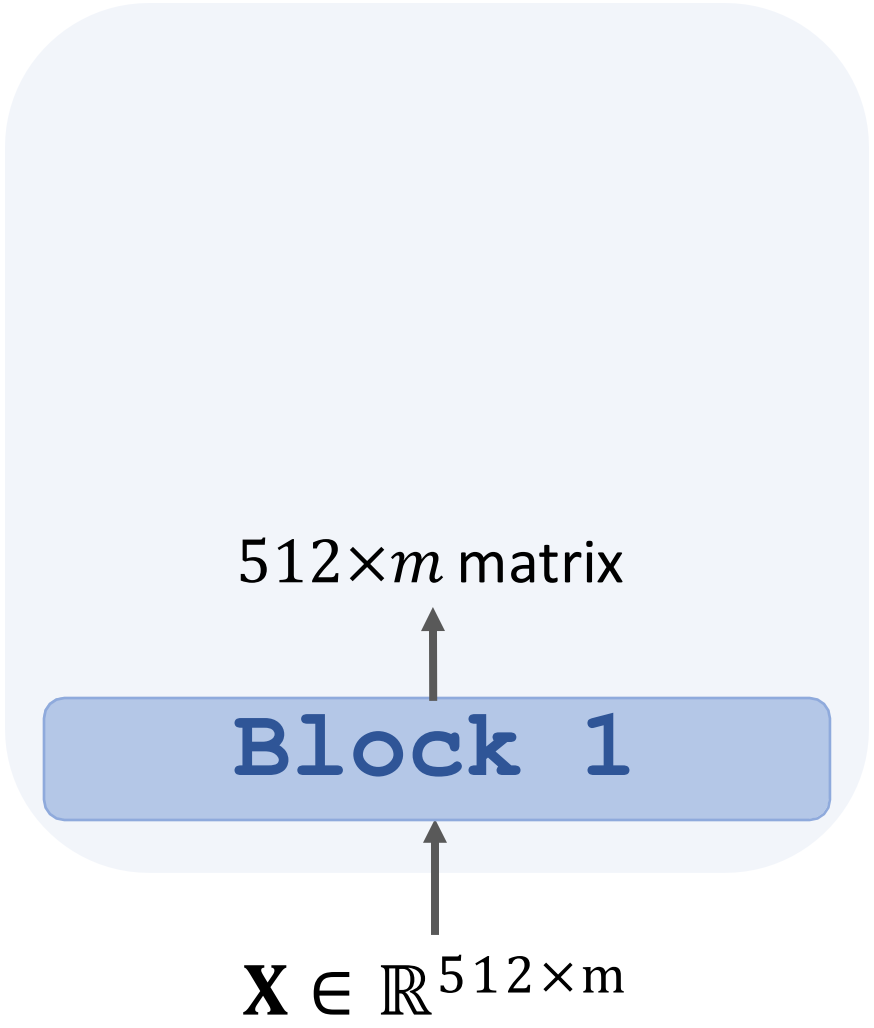


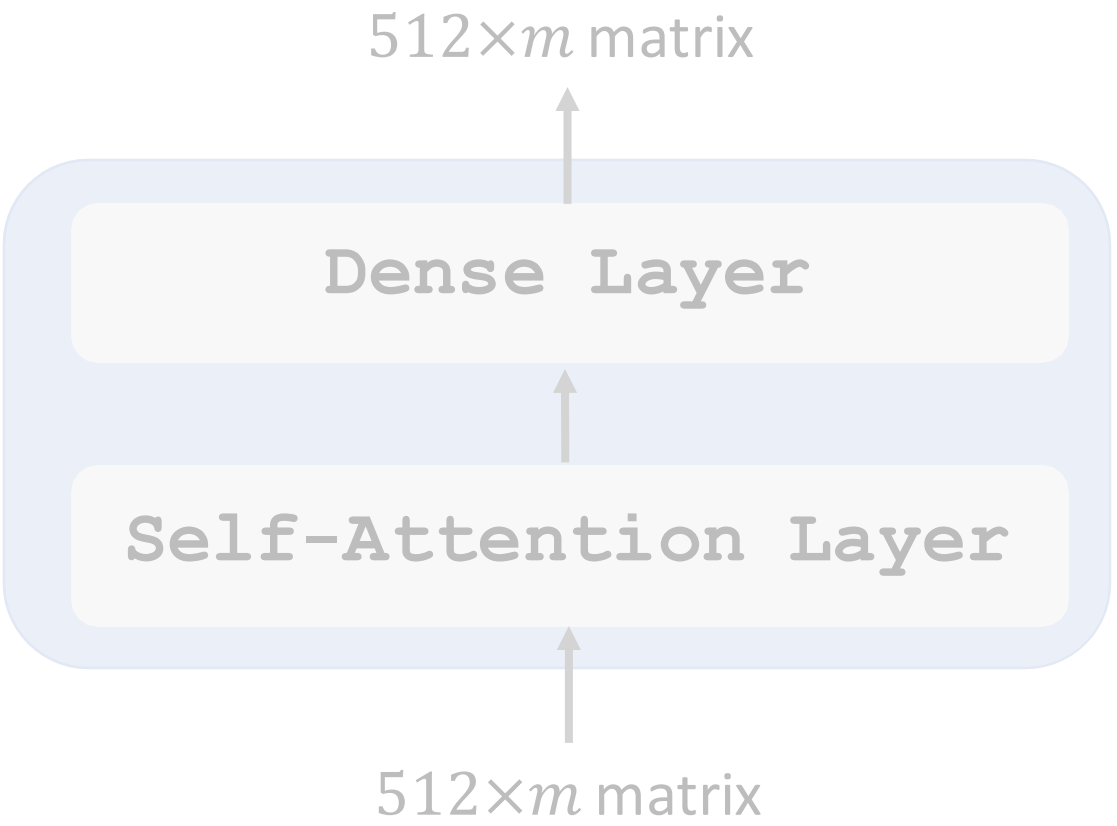
Transformerski Encoder



Jedan Blok Encodera

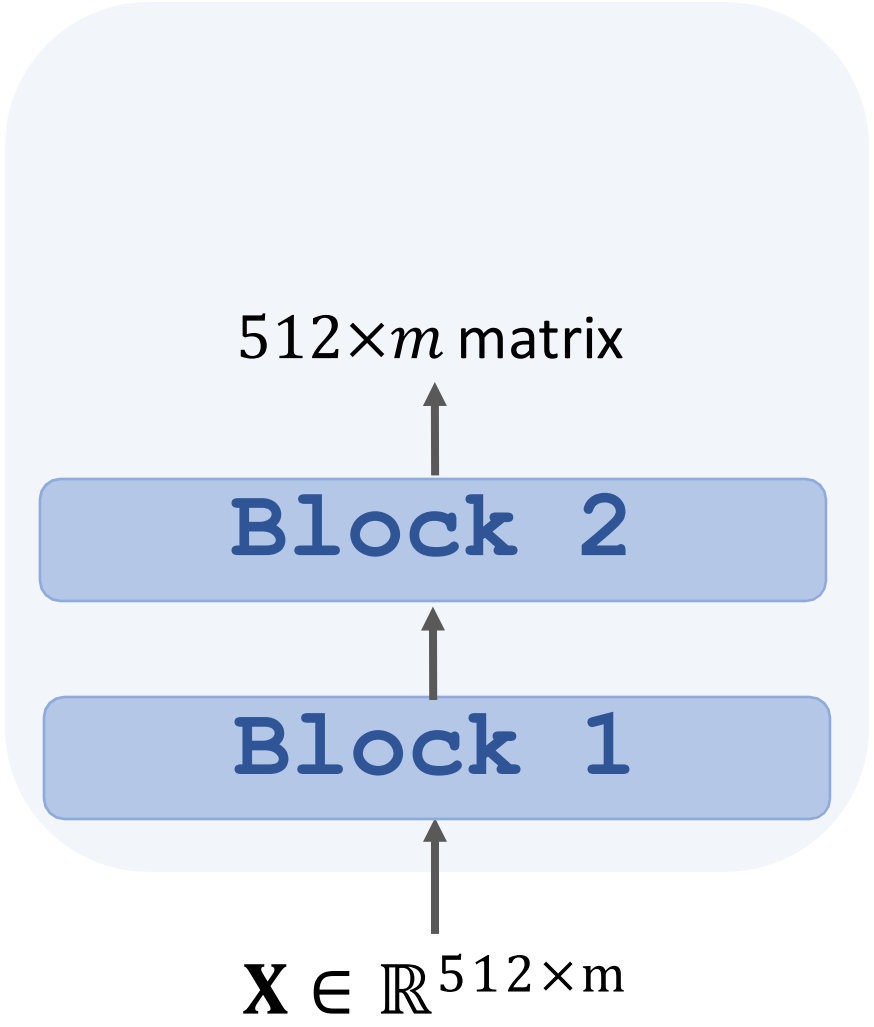
Transformerski Encoder

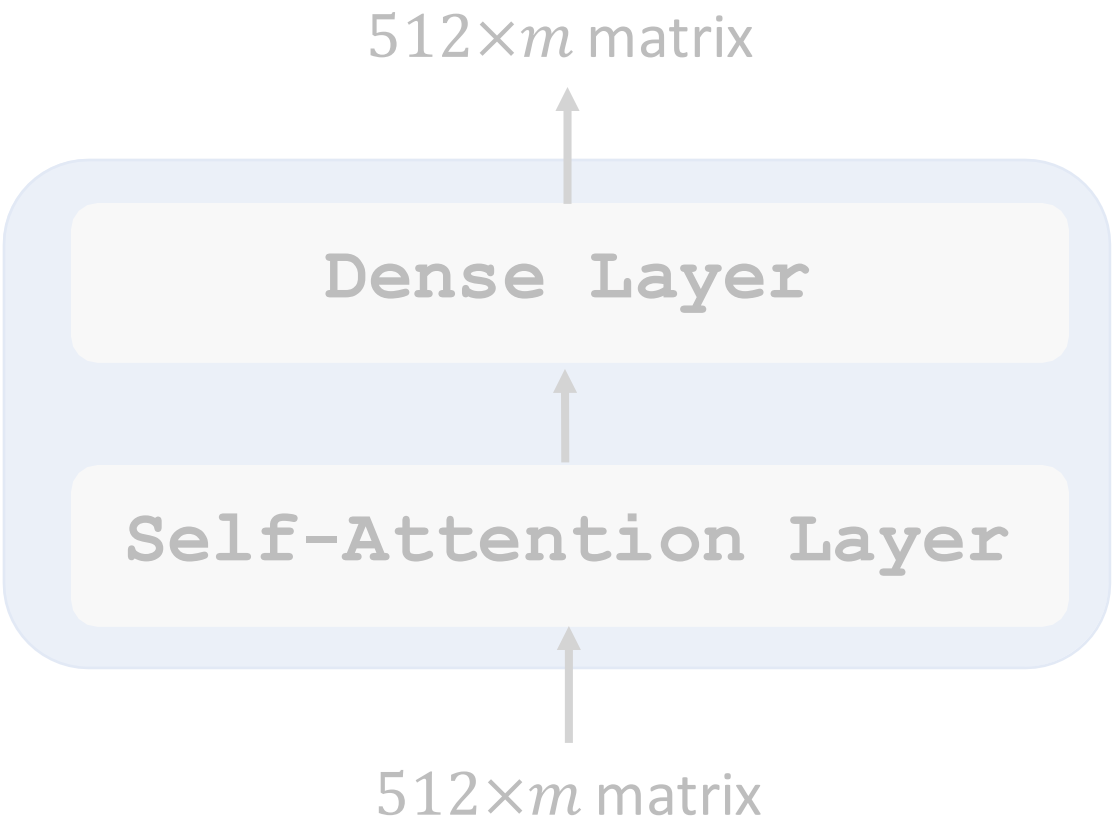




Jedan Blok Encodera

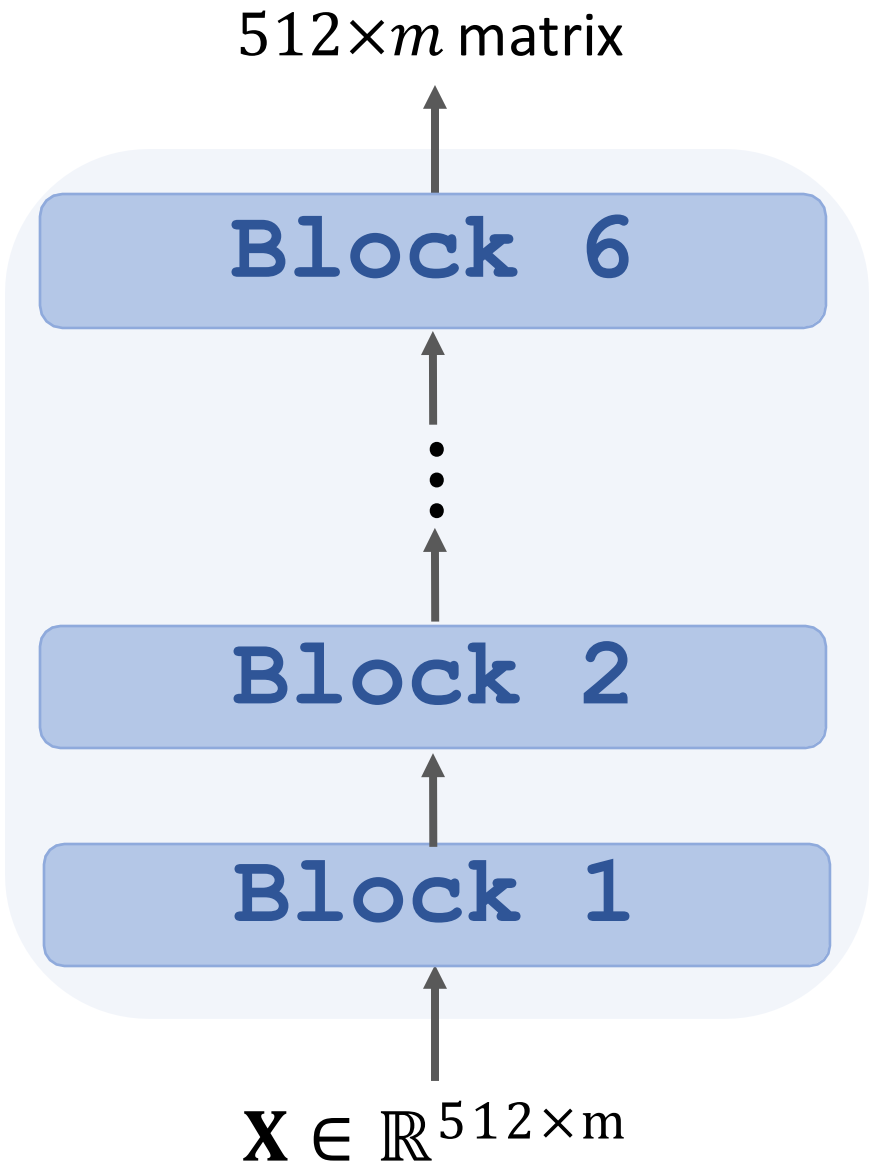
Transformerski Encoder

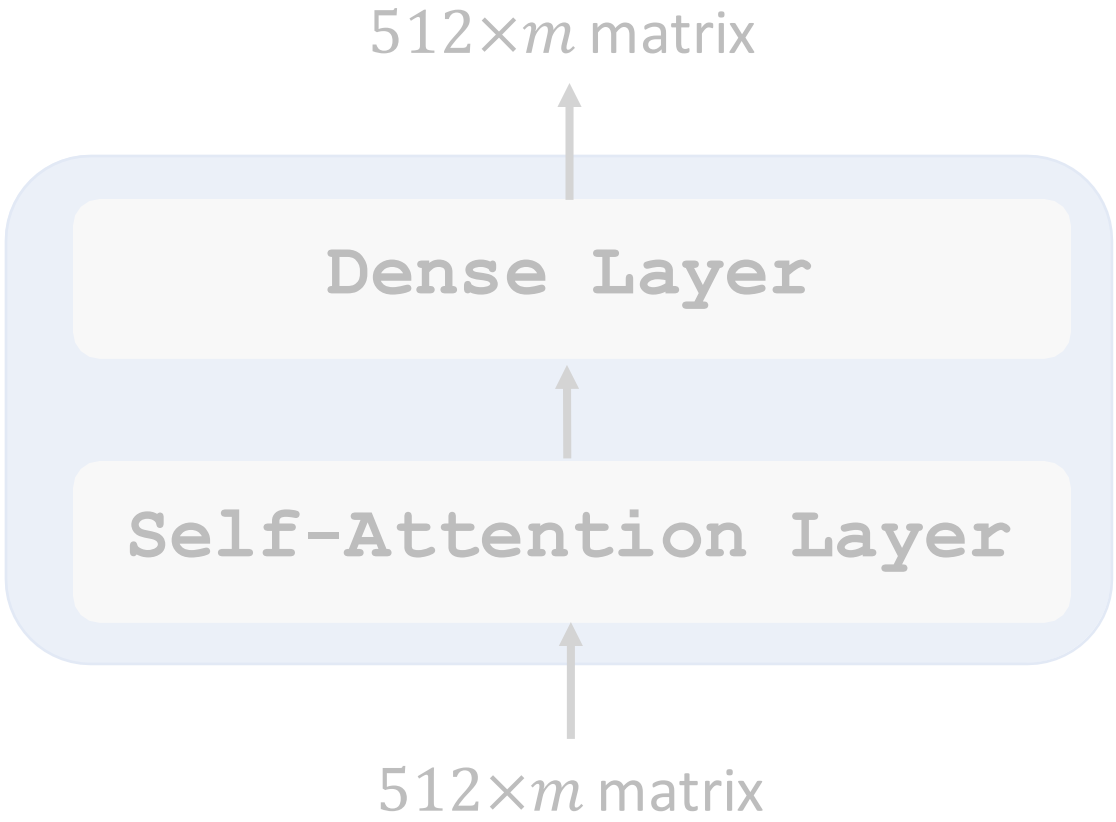




Jedan Blok Encodera

Transformerski Encoder



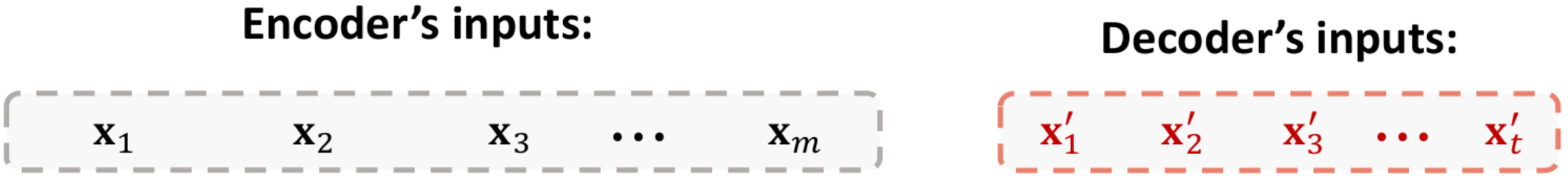


Jedan Blok Encodera

Stacked Attention slojevi

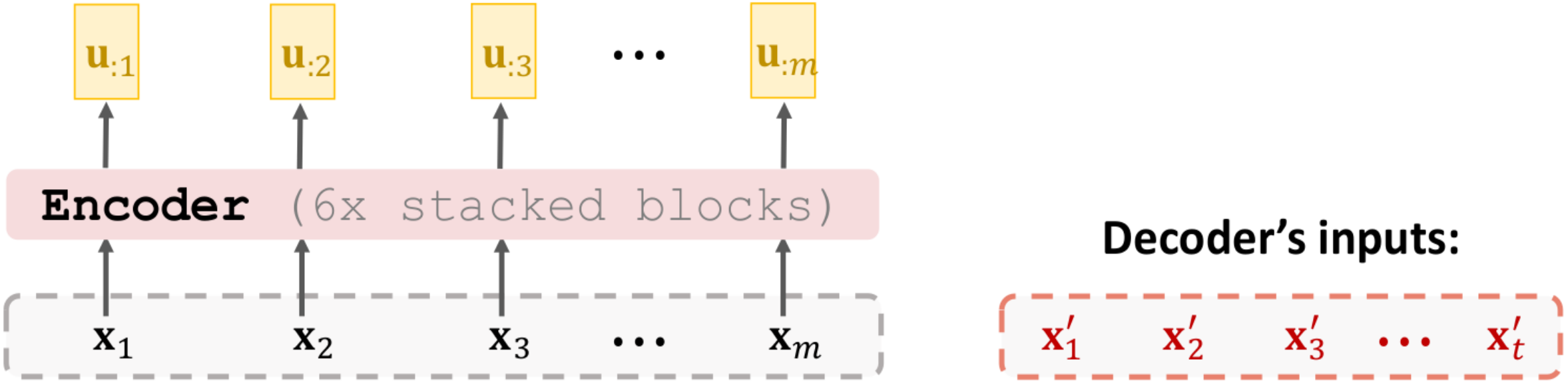
Stacked Attentions

- Transformer je Seq2Seq model (encoder + decoder).
- Encoderski ulazi vektori $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$.
- Decoderski ulazi su vektori $\mathbf{x}'_1, \mathbf{x}'_2, \cdots, \mathbf{x}'_t$.

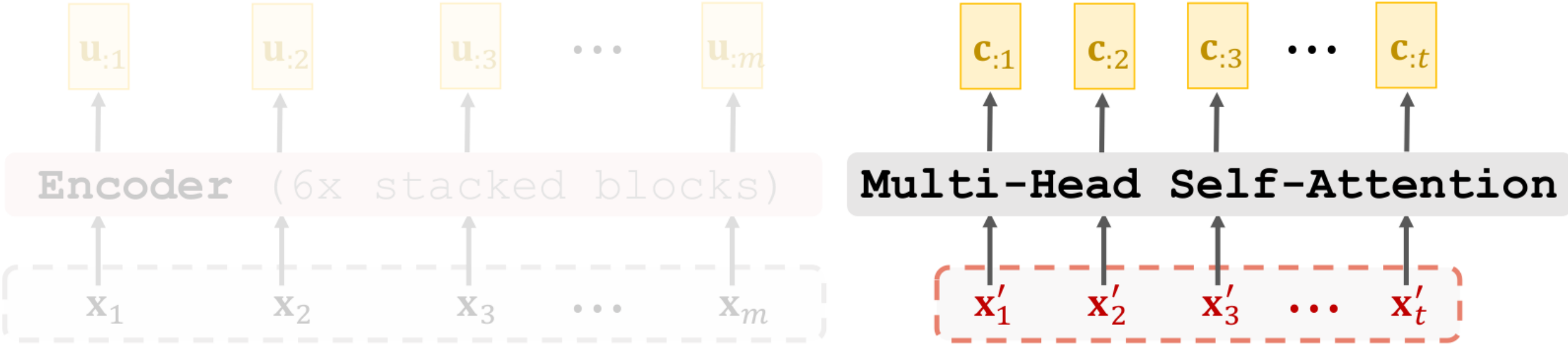


Stacked Attentions

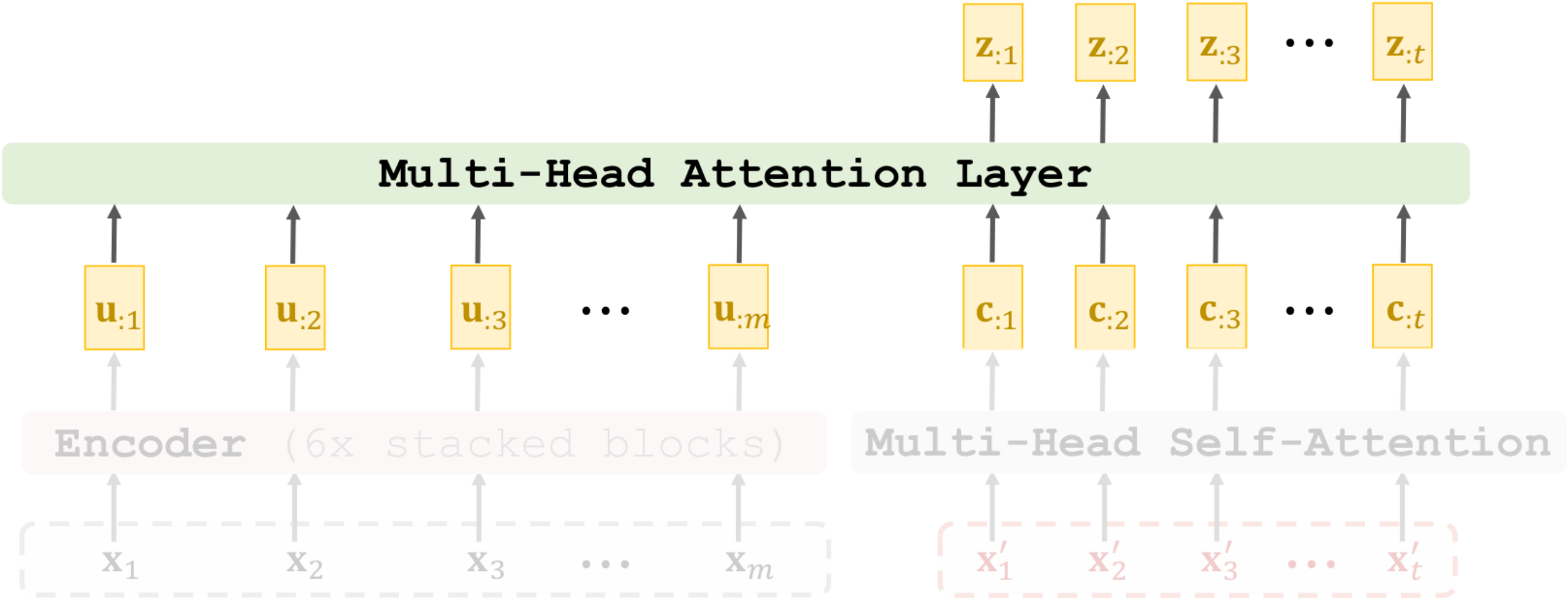
- Transformerski encoder sadrzi 6 stacked blocks.
- 1 block ≈ 1 multi-head attention layer + 1 dense layer.



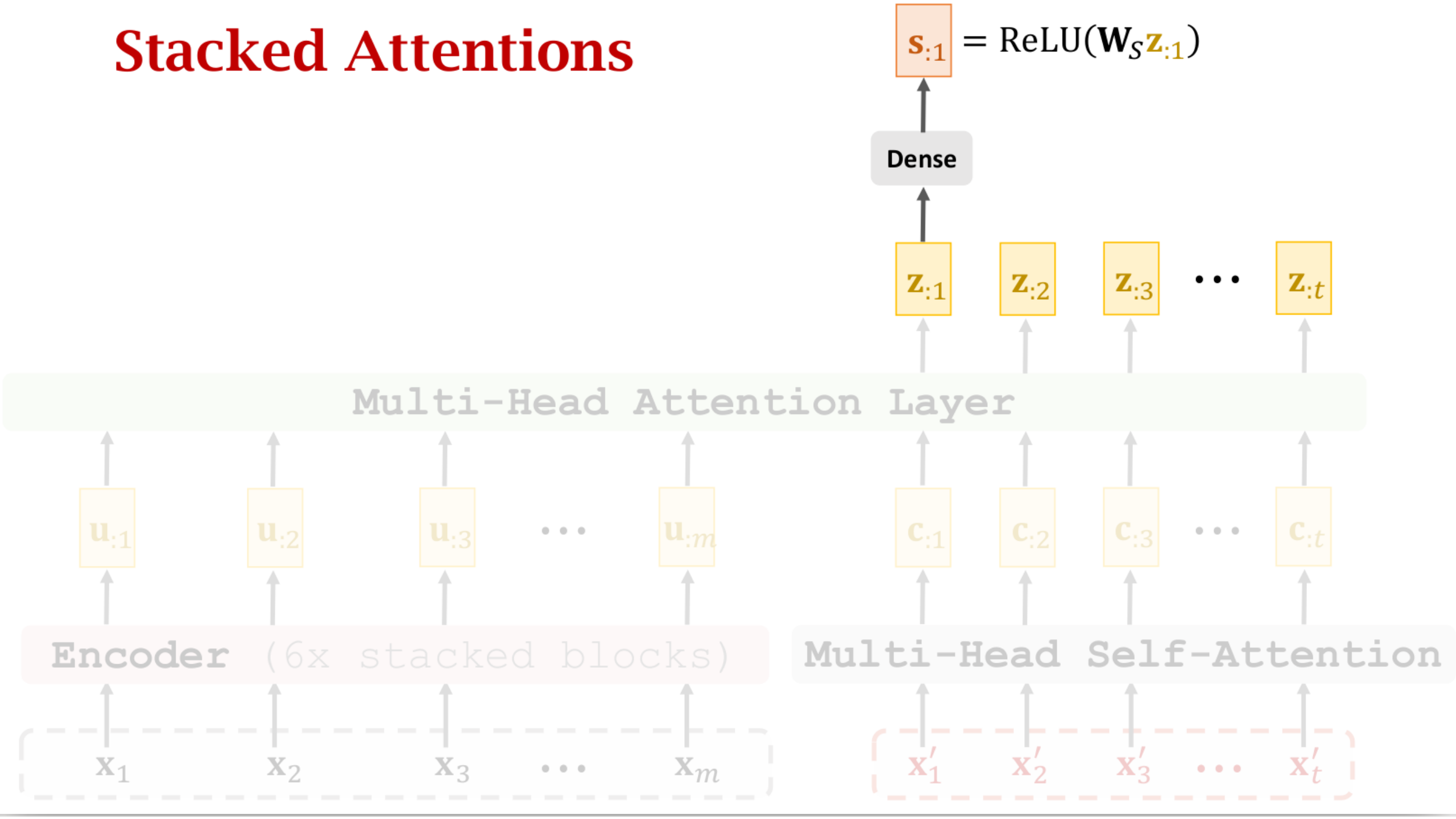
Stacked Attentions



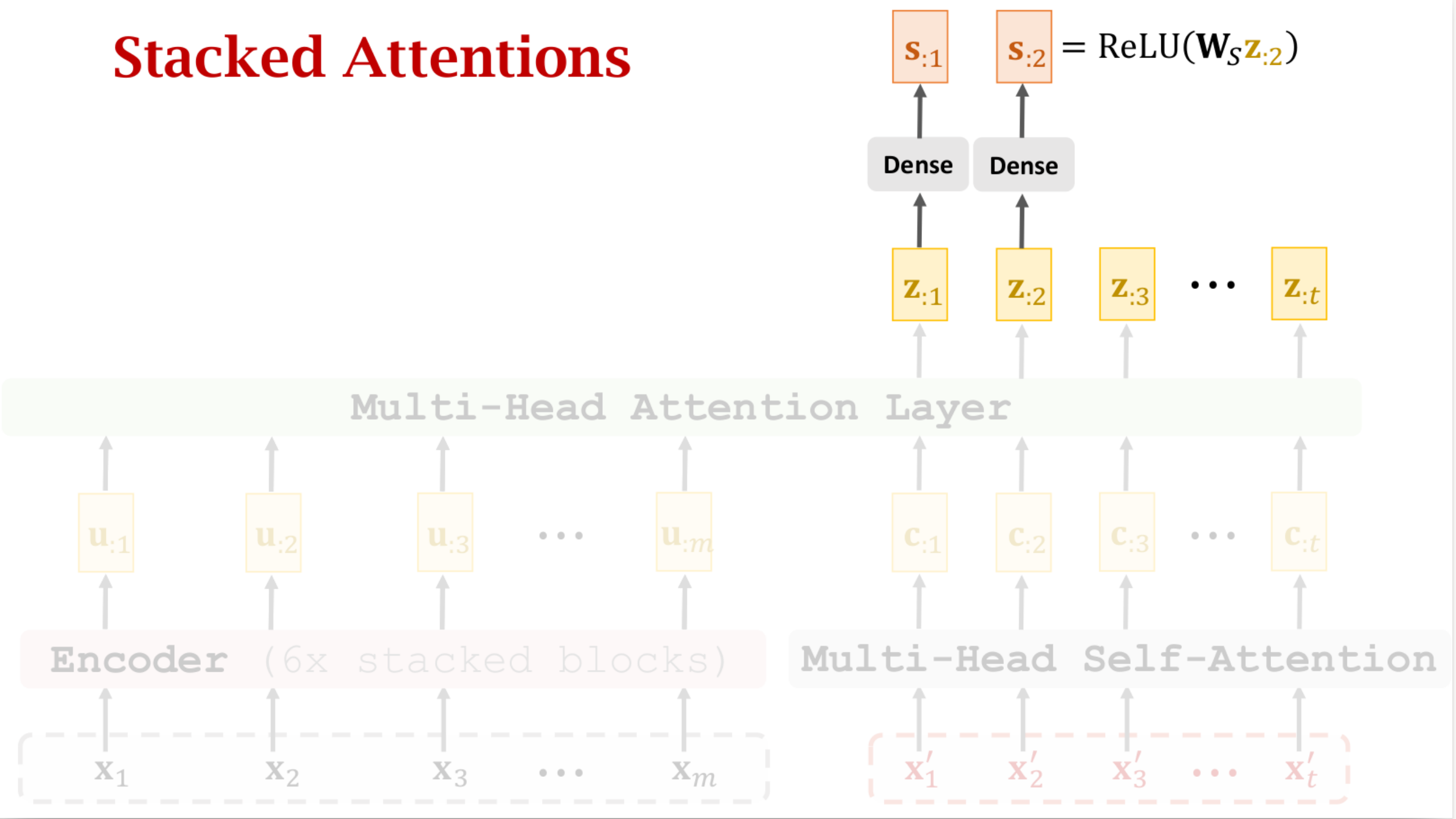
Stacked Attentions



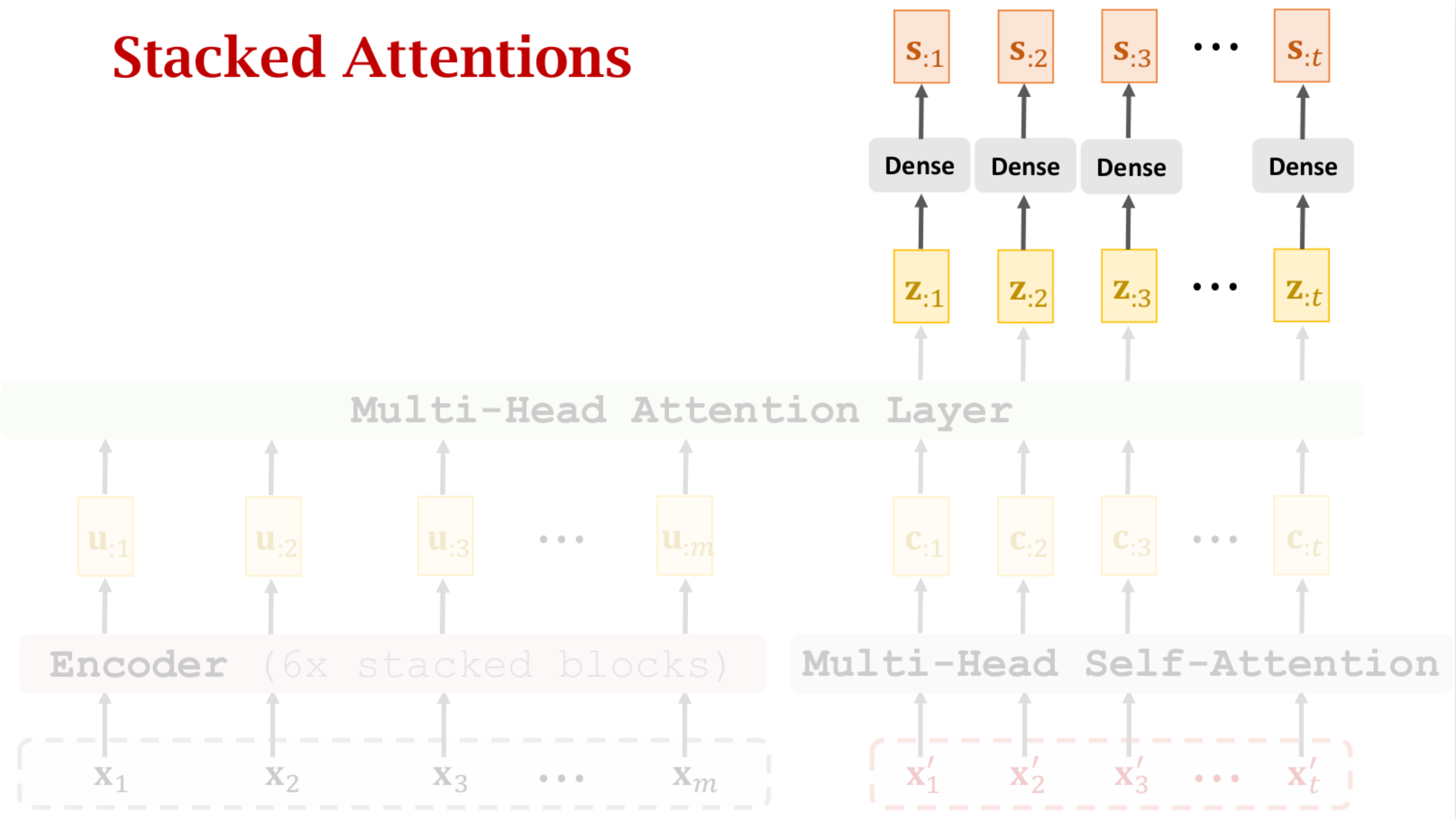
Stacked Attentions



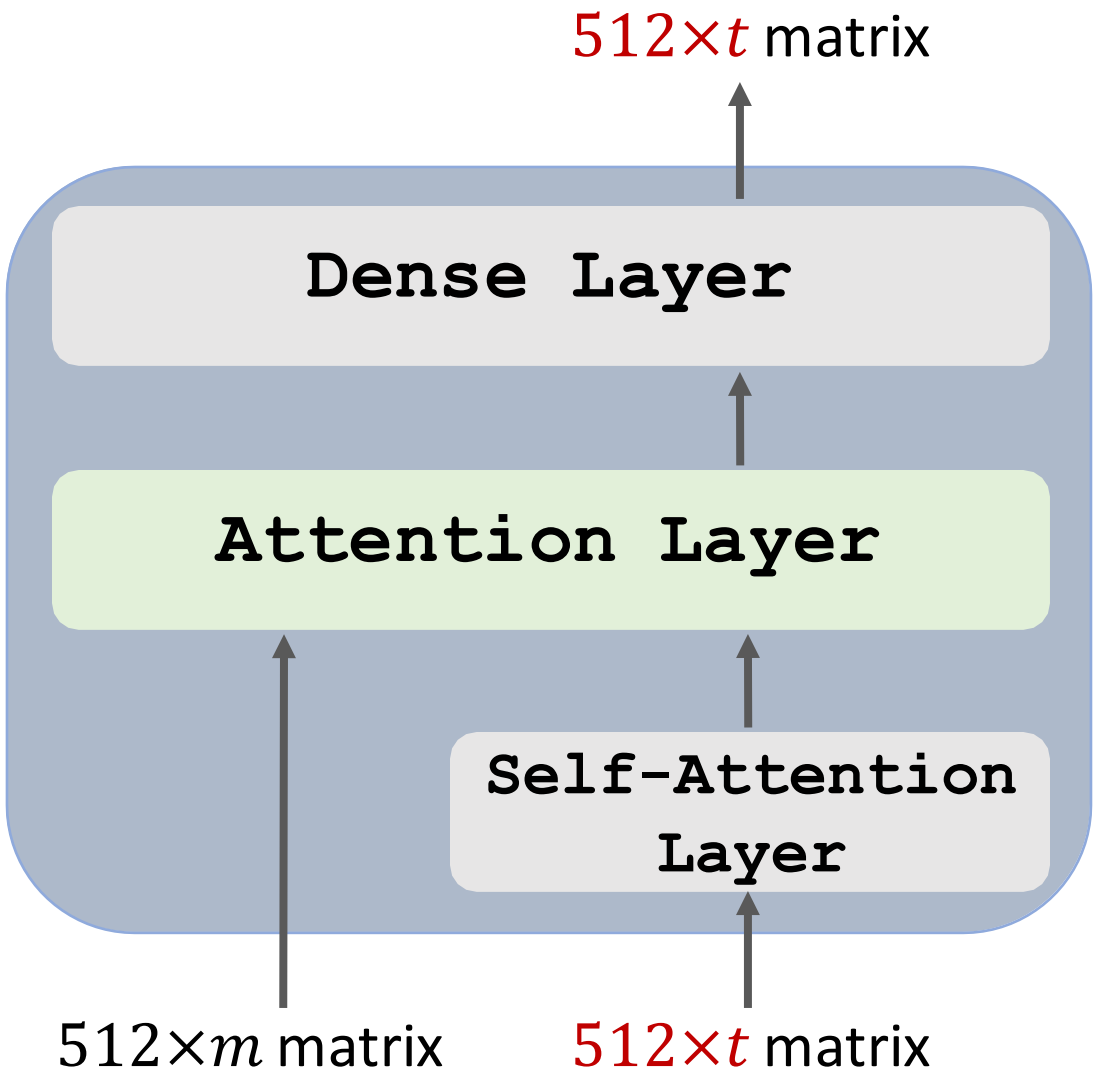
Stacked Attentions



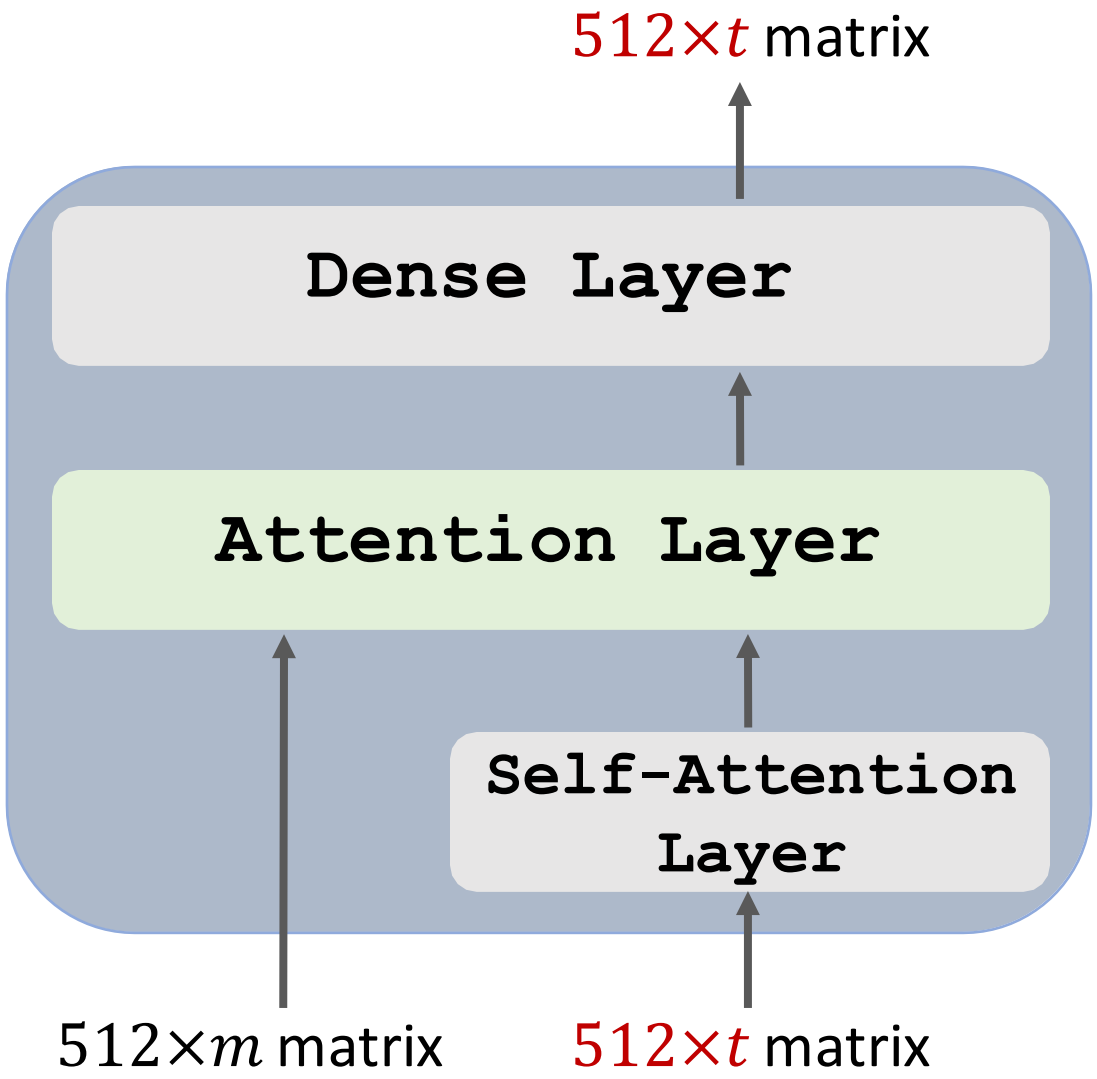
Stacked Attentions

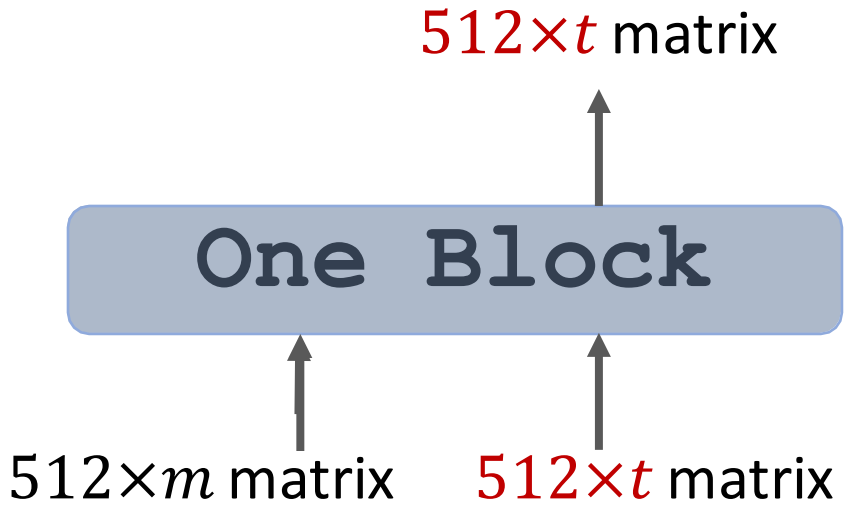


Transformerski Decoder : Jedan Blok



Transformerski Decoder : Jedan Blok

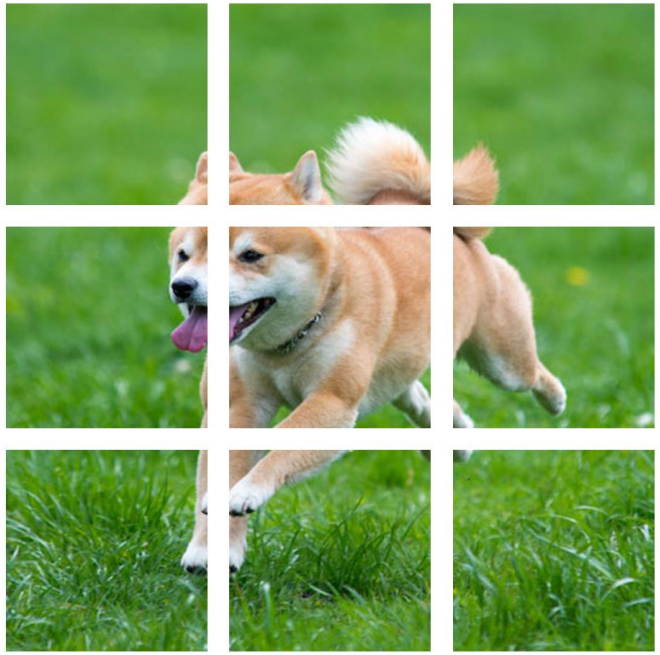




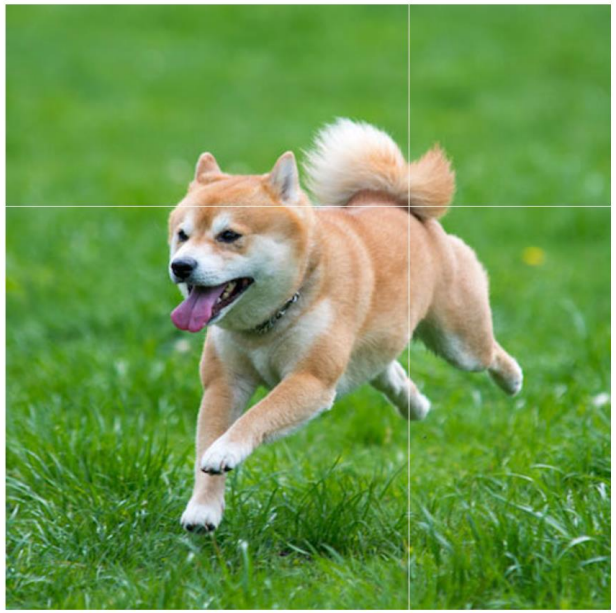
Vision Transformer (ViT)

http://wangshusen.github.io/

Vektorizacija

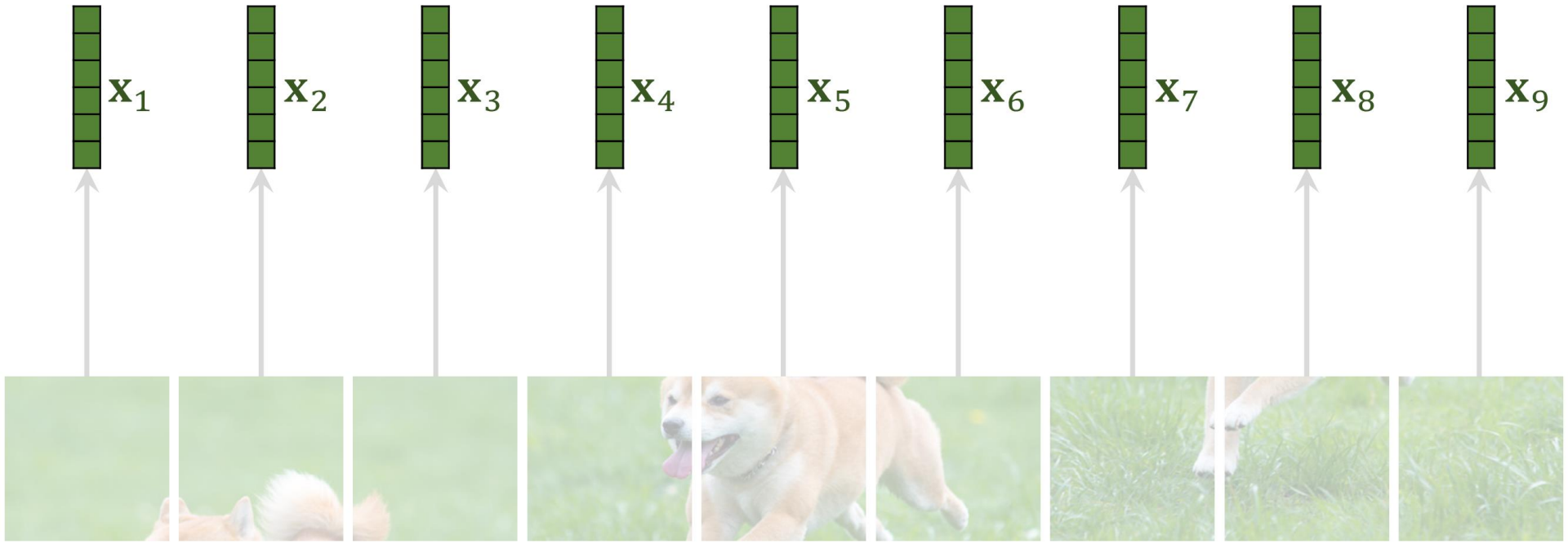


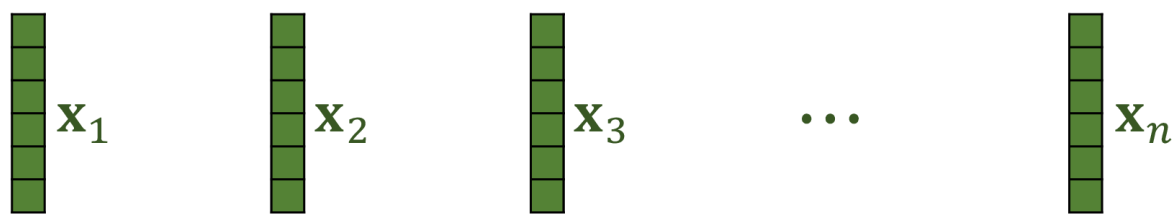
Vektorizacija

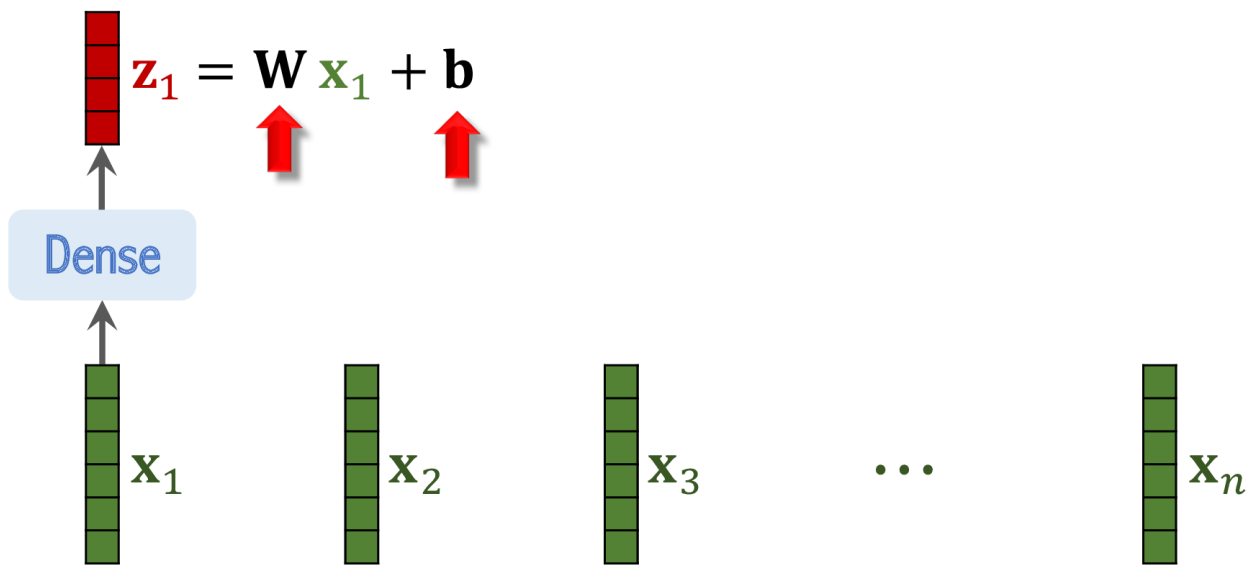


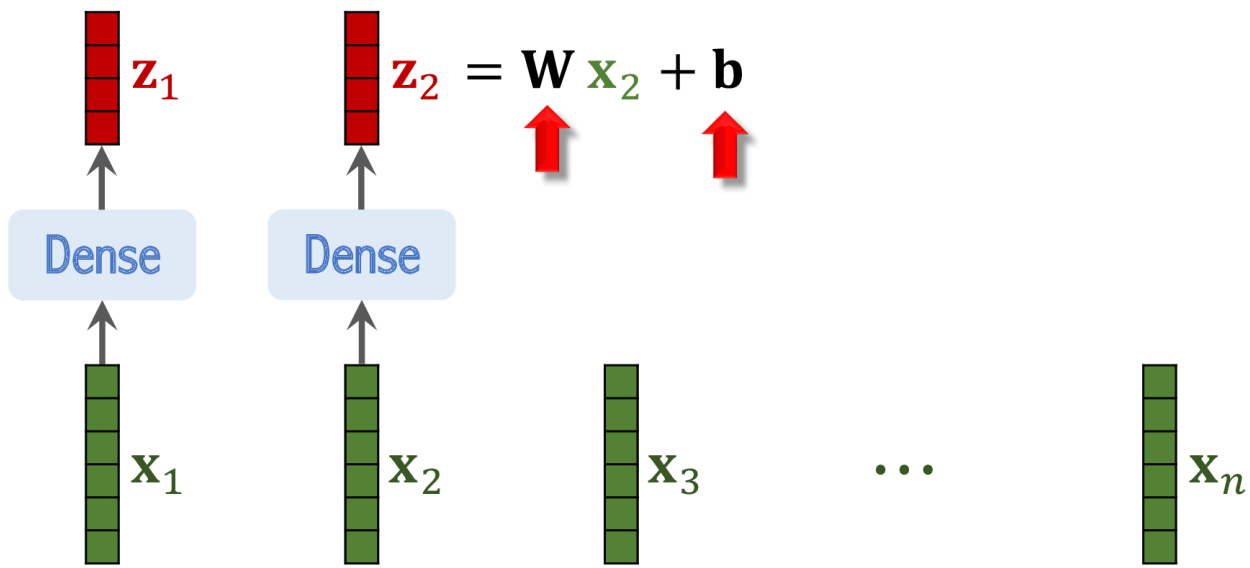
Vektorizacija

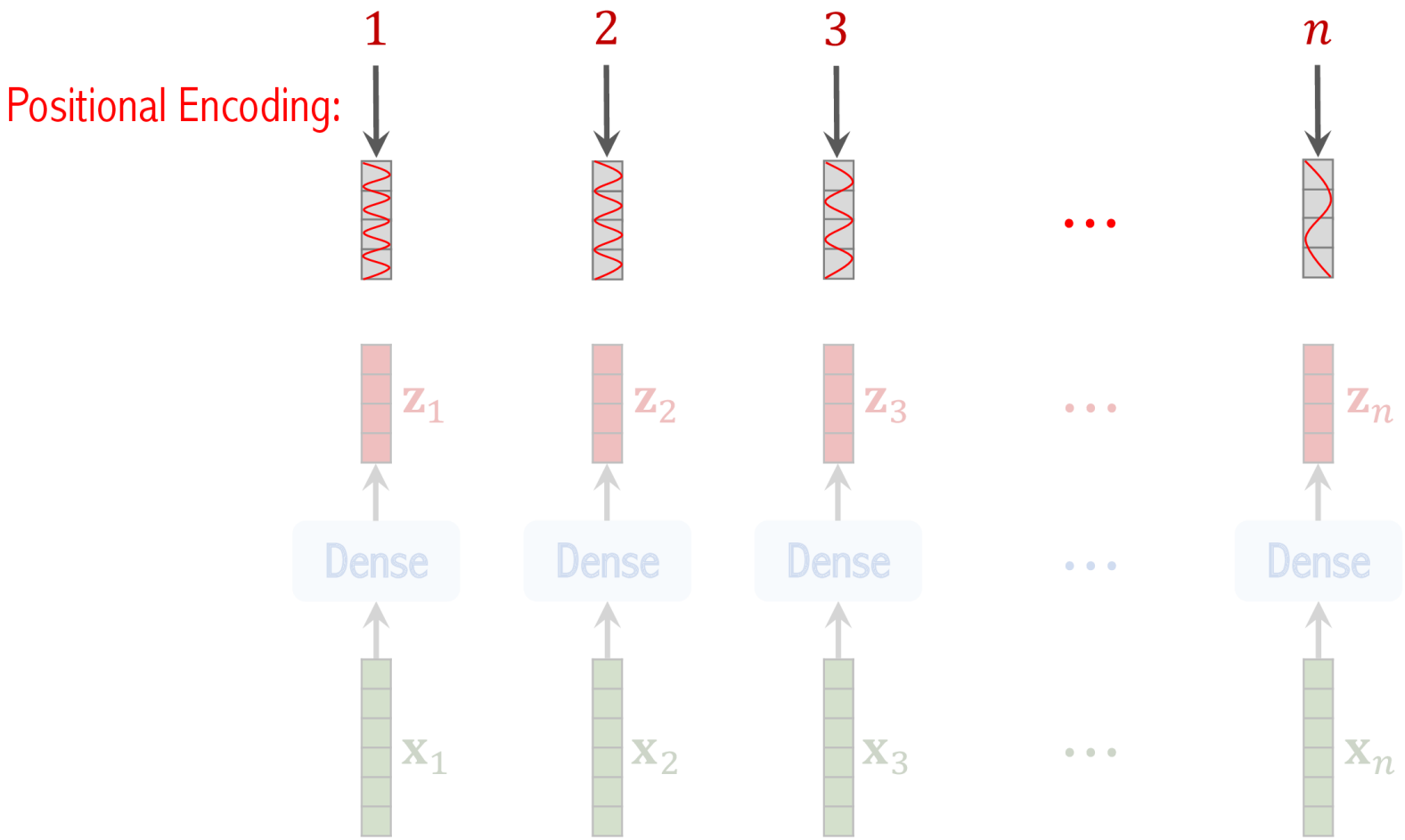
If the patches are $d_1 \times d_2 \times d_3$ tensors, then the vectors are $d_1 d_2 d_3 \times 1$.

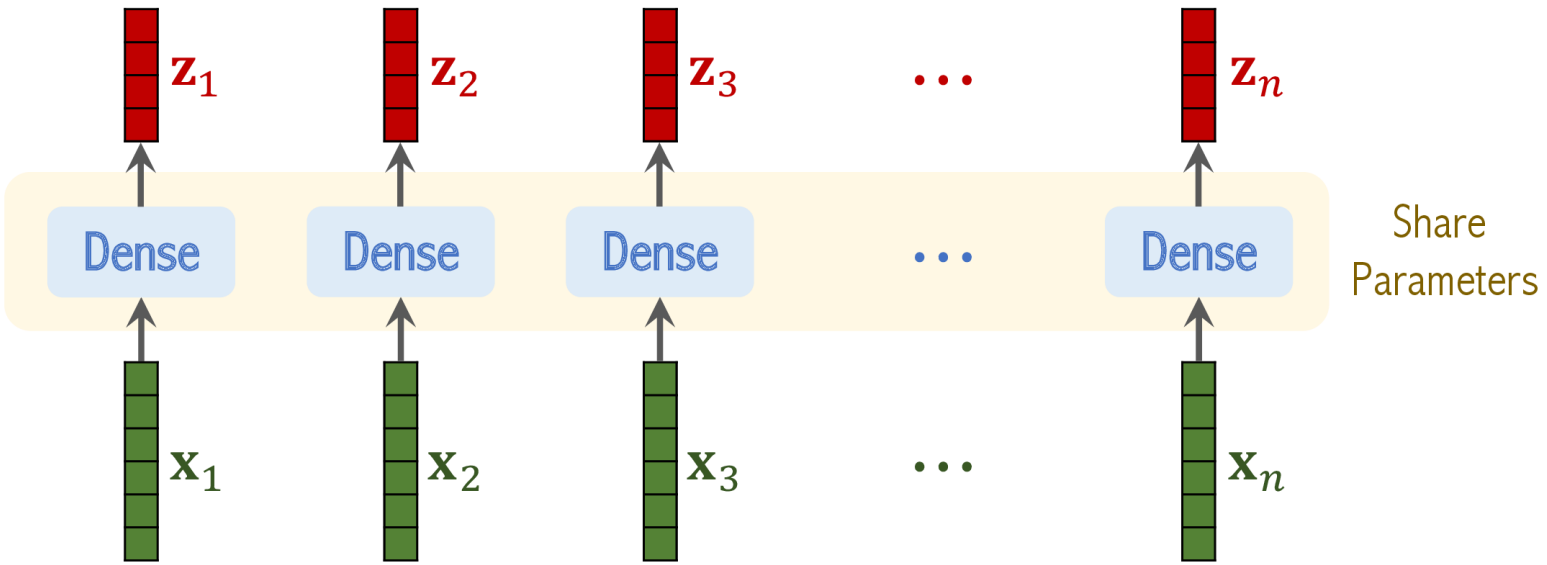



$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\cdots$
$\mathbf{x}_n$

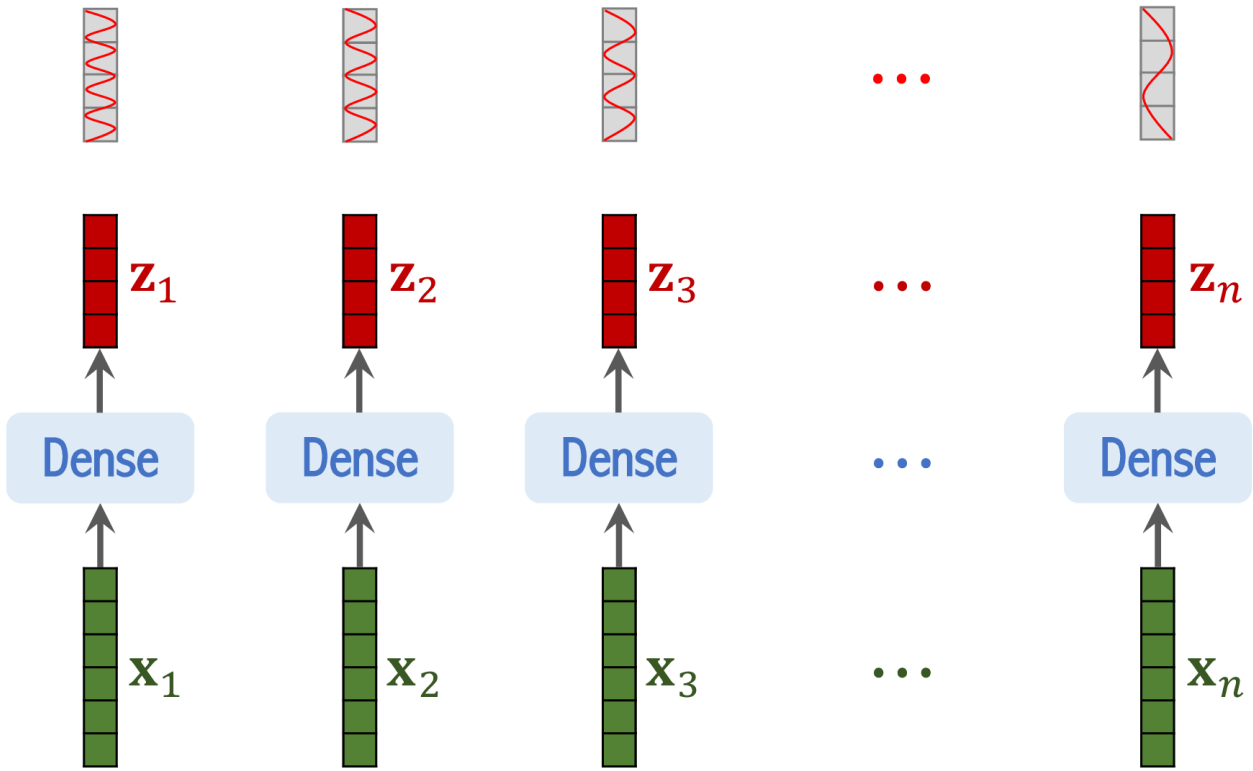

$\mathbf{z}_1 = \mathbf{W}\,\mathbf{x}_1 + \mathbf{b}$
Dense
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\cdots$
$\mathbf{x}_n$


$\mathbf{z}_1$
$\mathbf{z}_2 = \mathbf{W}\,\mathbf{x}_2 + \mathbf{b}$
Dense
Dense
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\cdots$
$\mathbf{x}_n$


Positional Encoding:
1
2
3
n
Dense
Dense
Dense
Dense
z_1
z_2
z_3
z_n
x_1
x_2
x_3
x_n

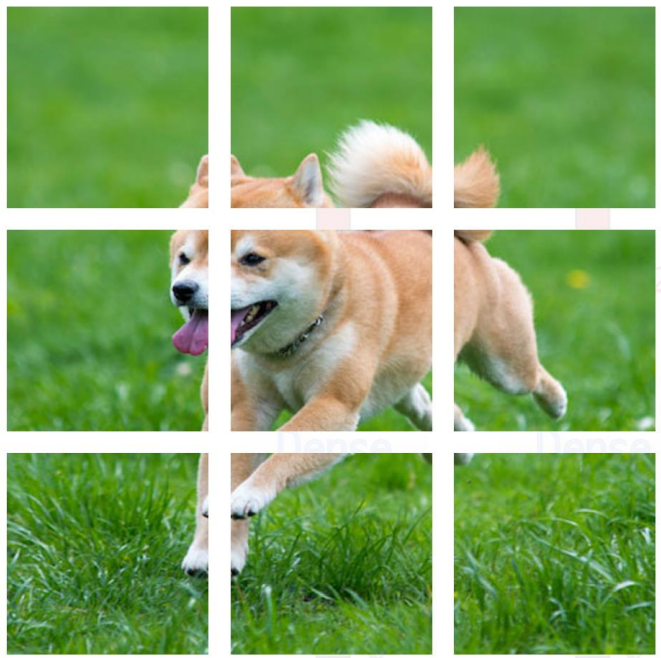

$\mathbf{z}_1$
$\mathbf{z}_2$
$\mathbf{z}_3$
$\mathbf{z}_n$
Dense
Dense
Dense
Dense
Share Parameters
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_n$

Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$.

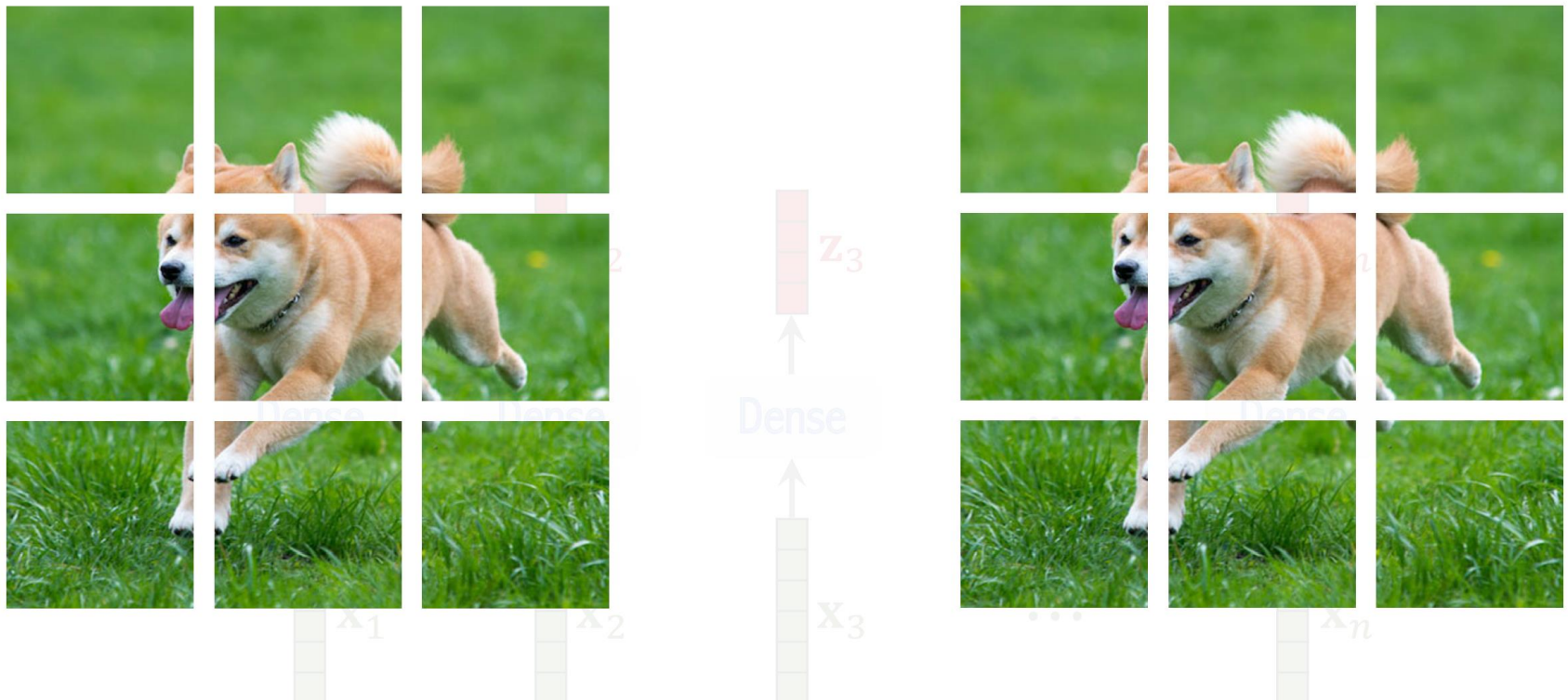


Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$. (Why?)

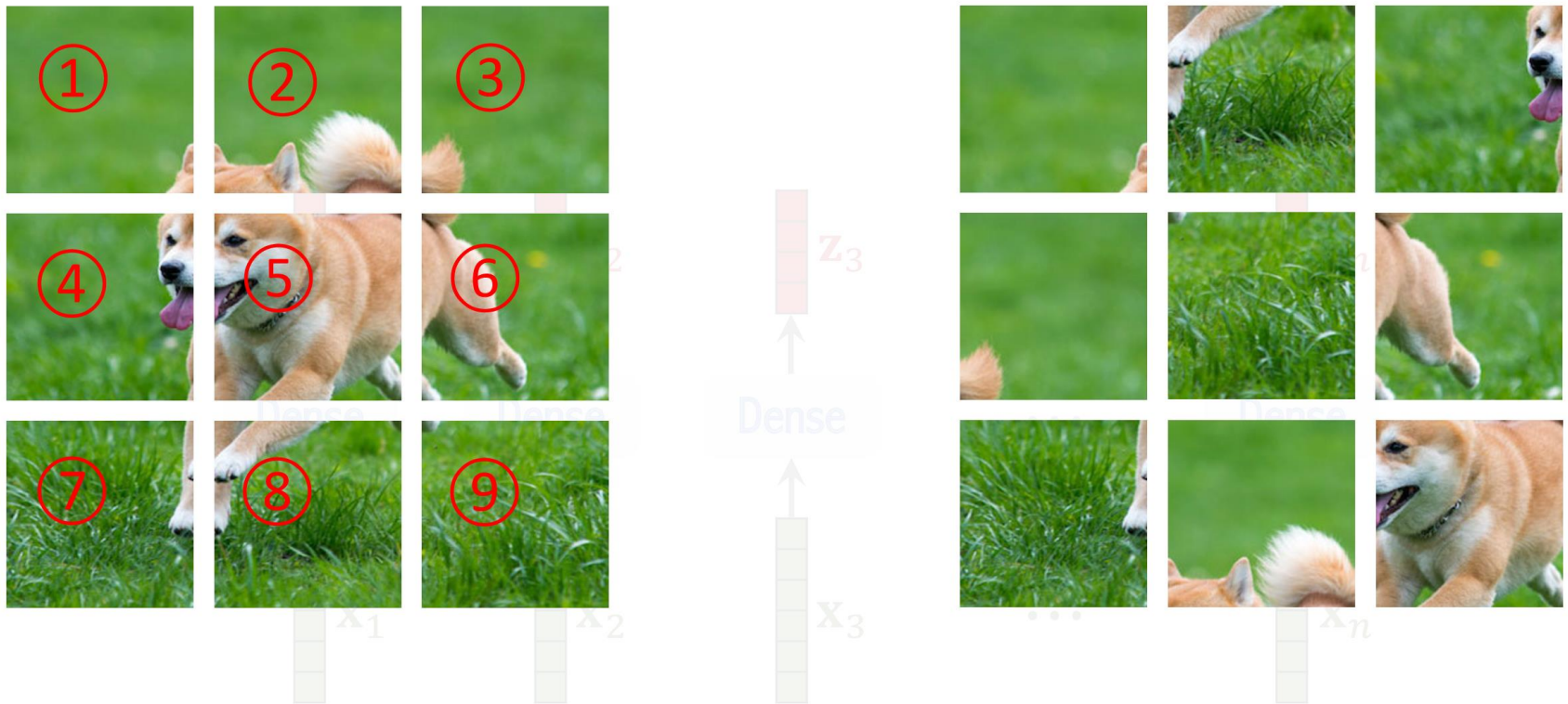
Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$. (Why?)

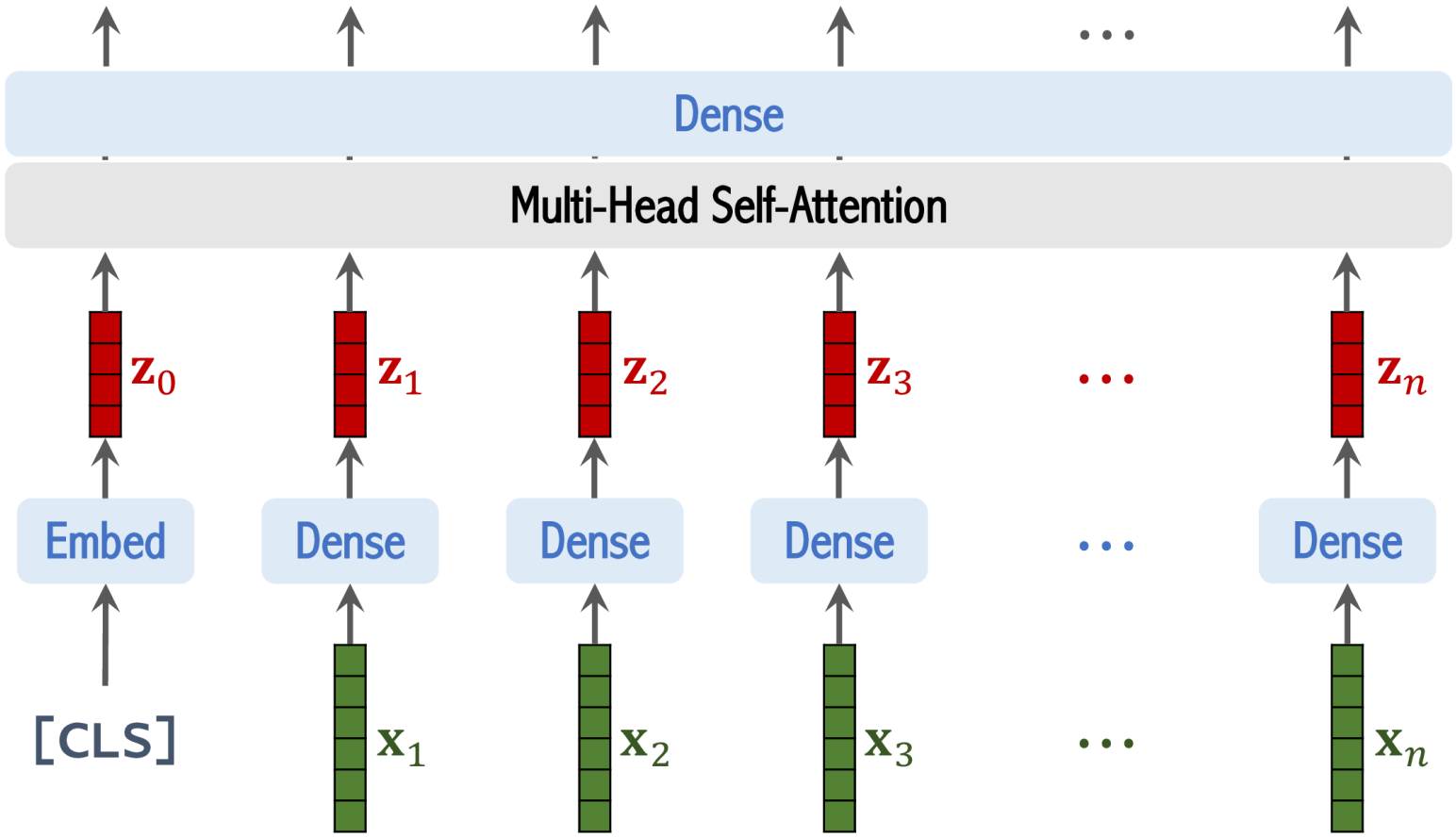


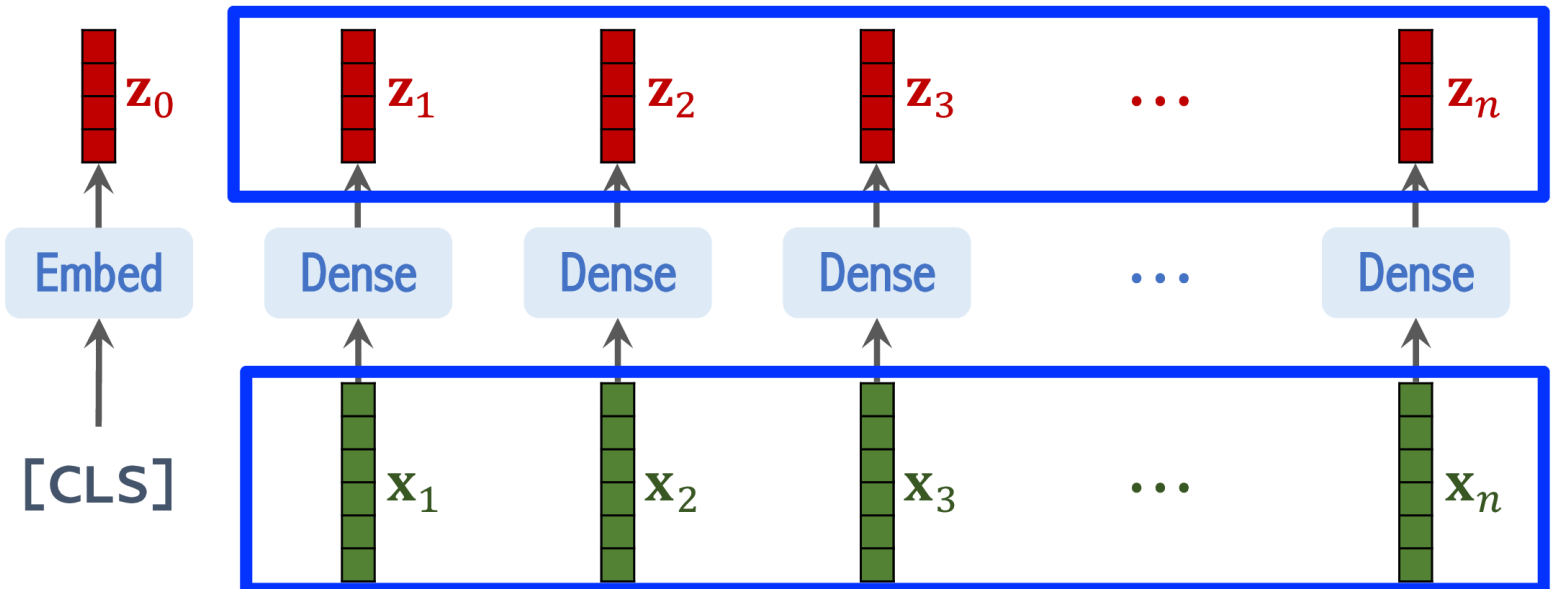
Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$. (Why?)

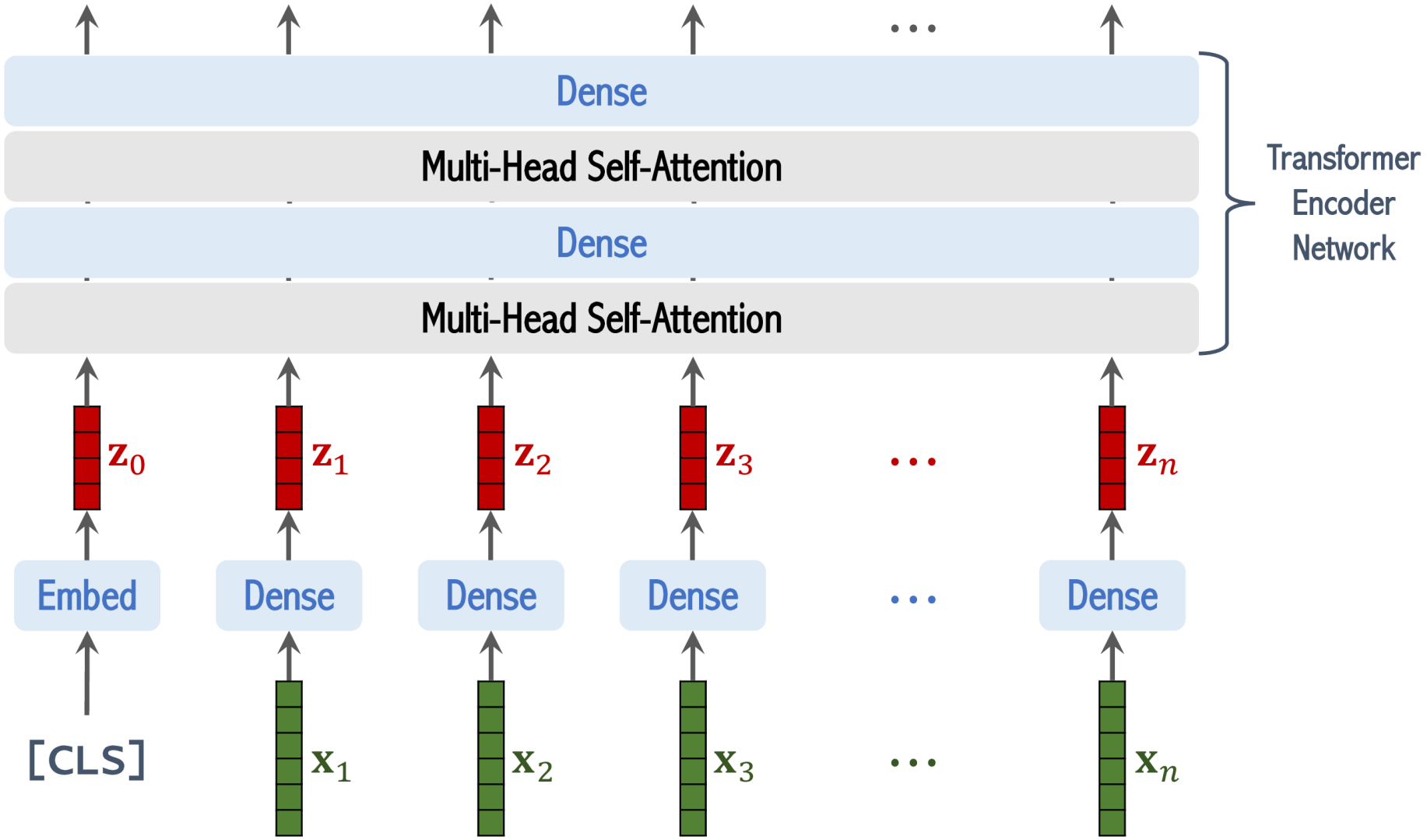


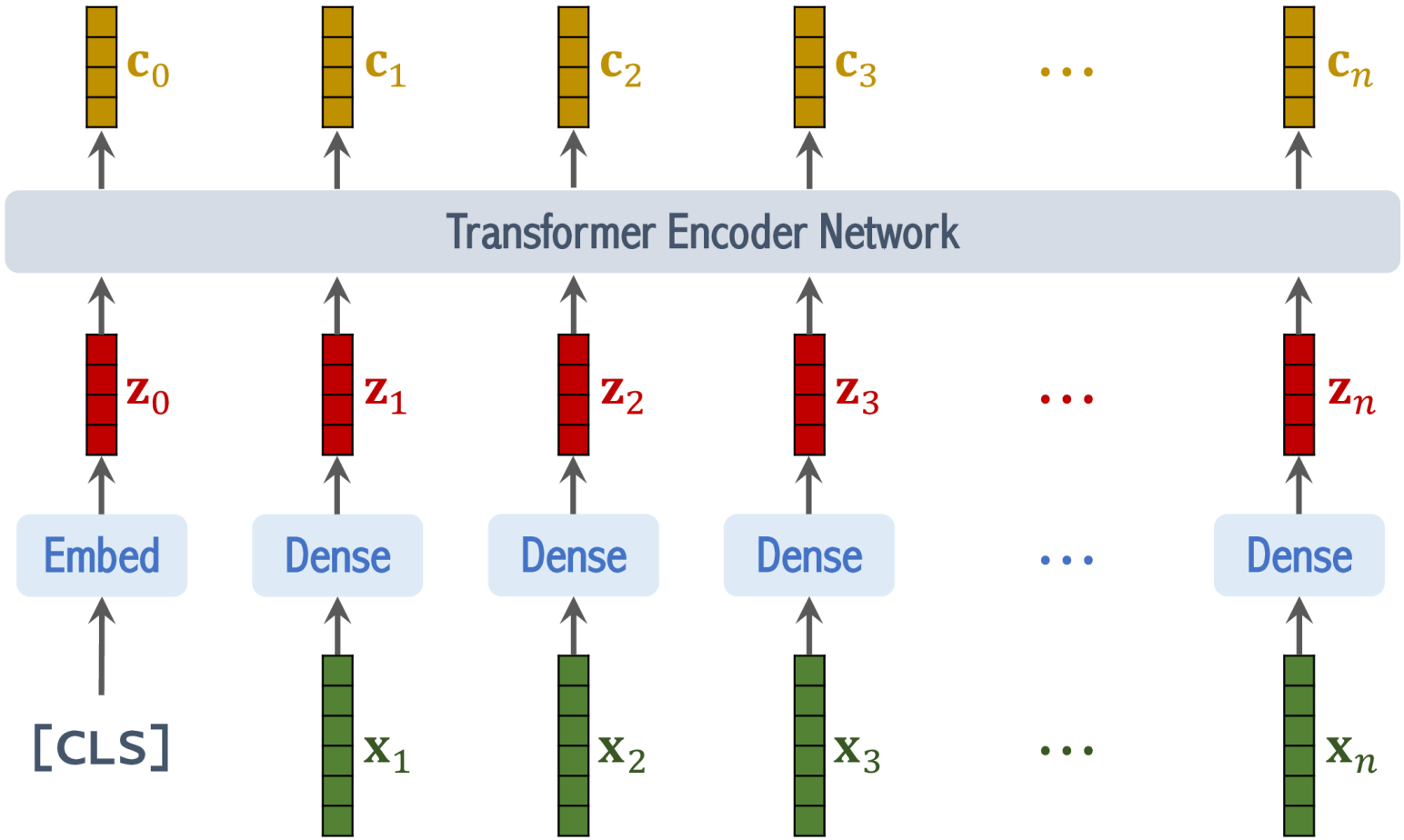
Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$. (Why?)

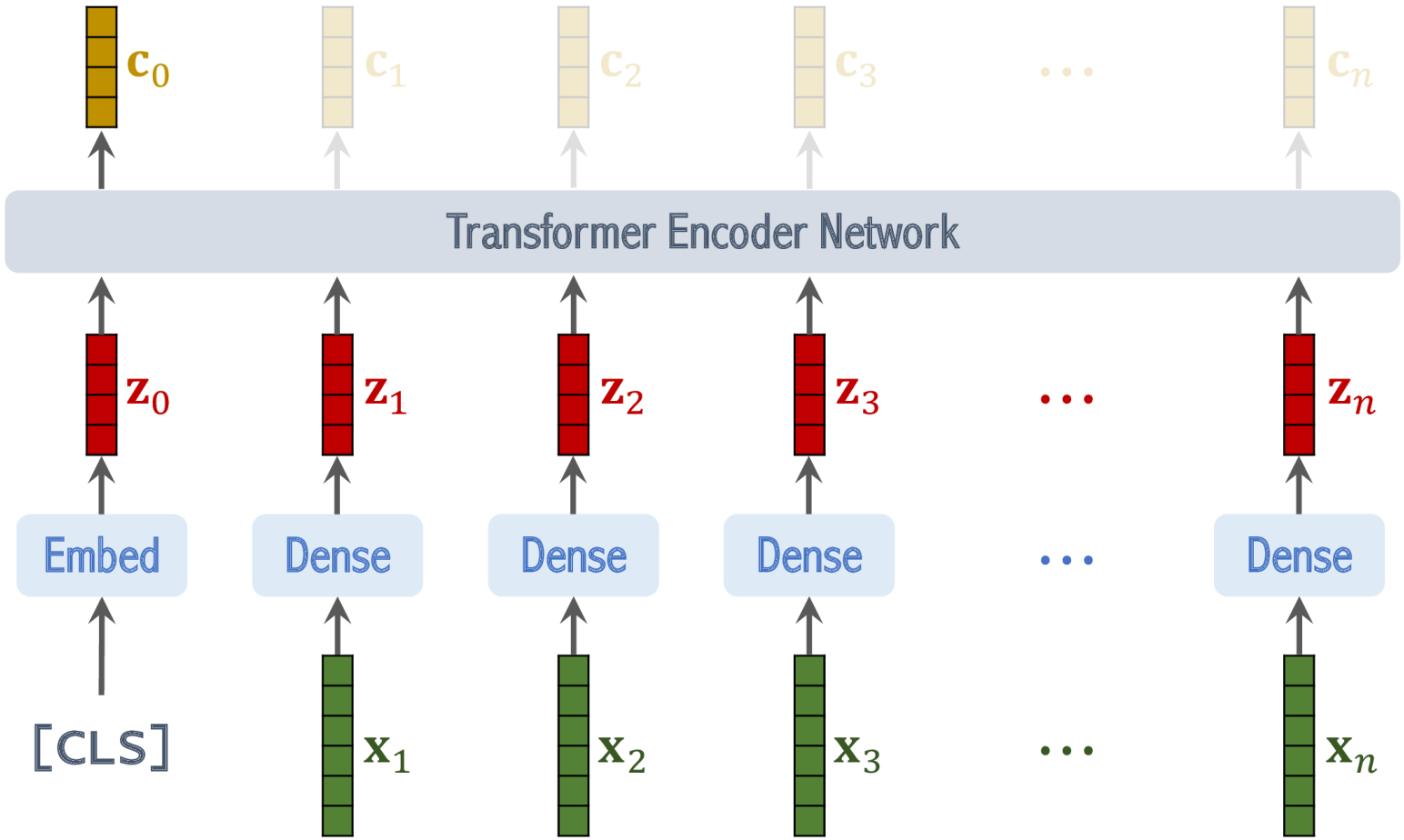


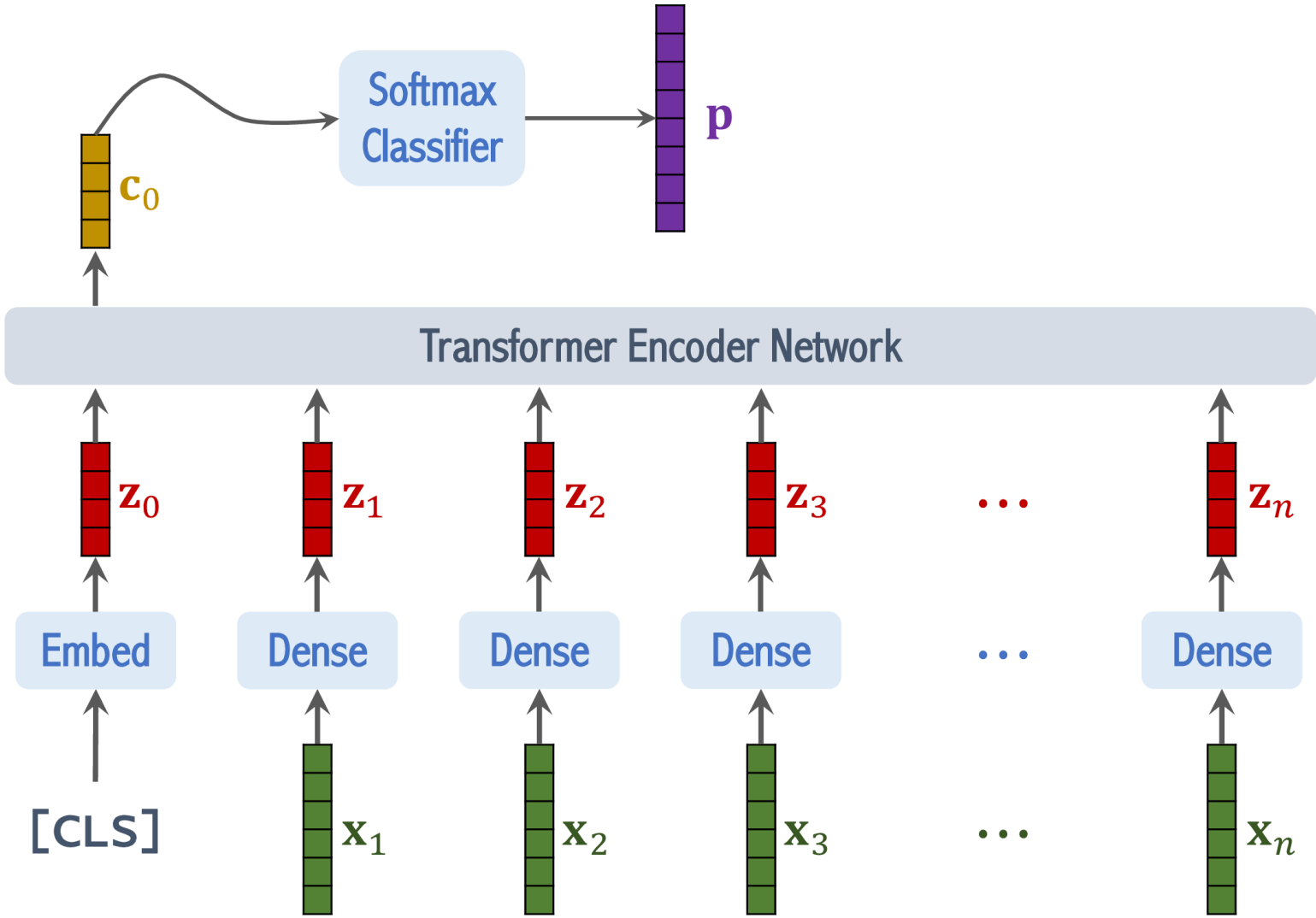

Dense
Multi-Head Self-Attention
$\mathbf{z}_0$
$\mathbf{z}_1$
$\mathbf{z}_2$
$\mathbf{z}_3$
$\mathbf{z}_n$
Embed
Dense
Dense
Dense
Dense
[CLS]
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_n$

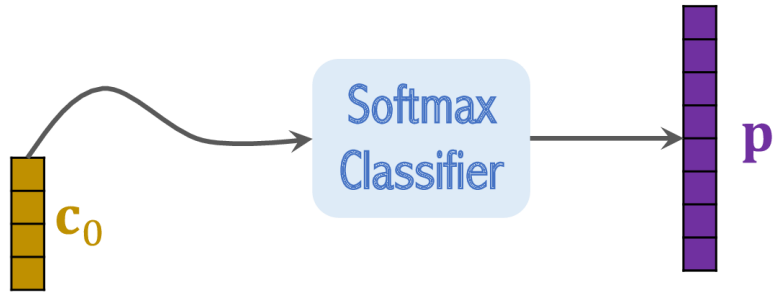

$\mathbf{z}_0$
$\mathbf{z}_1$
$\mathbf{z}_2$
$\mathbf{z}_3$
• • •
$\mathbf{z}_n$
Embed
Dense
Dense
Dense
• • •
Dense
[CLS]
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
• • •
$\mathbf{x}_n$

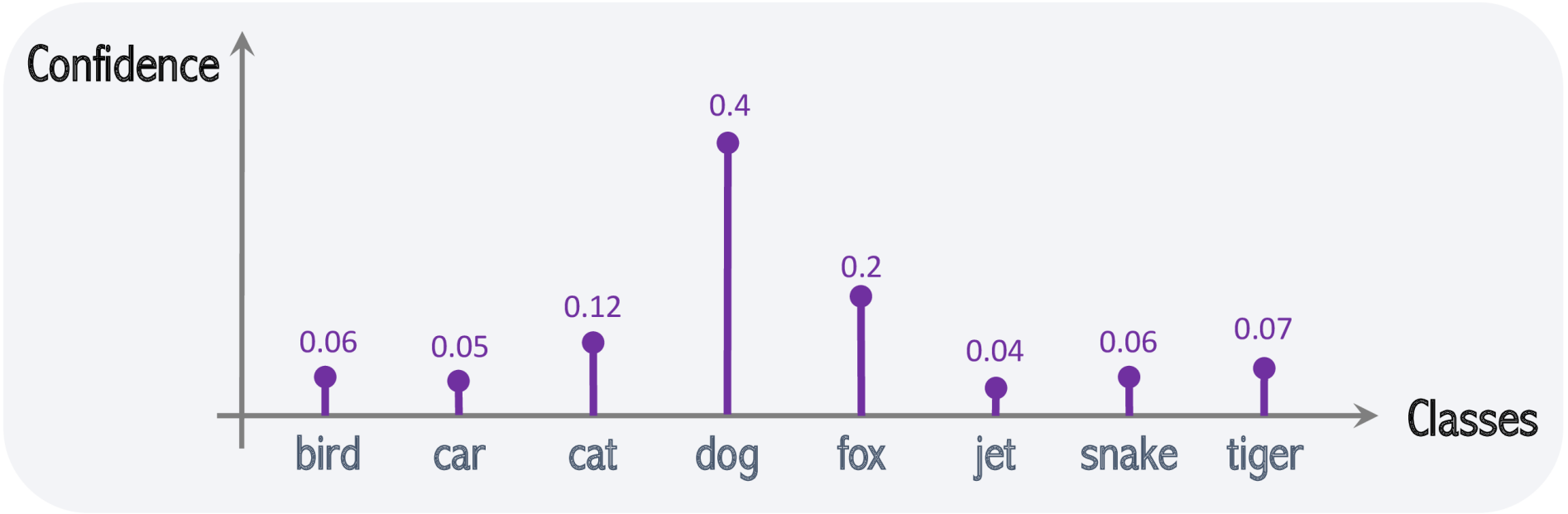

Dense
Multi-Head Self-Attention
Dense
Multi-Head Self-Attention
Transformer Encoder Network
z_0
z_1
z_2
z_3
z_n
Embed
Dense
Dense
Dense
Dense
[CLS]
x_1
x_2
x_3
x_n


$\mathbf{c}_0$
$\mathbf{c}_1$
$\mathbf{c}_2$
$\mathbf{c}_3$
$\cdots$
$\mathbf{c}_n$
Transformer Encoder Network
$\mathbf{z}_0$
$\mathbf{z}_1$
$\mathbf{z}_2$
$\mathbf{z}_3$
$\cdots$
$\mathbf{z}_n$
Embed
Dense
Dense
Dense
$\cdots$
Dense
[CLS]
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\cdots$
$\mathbf{x}_n$

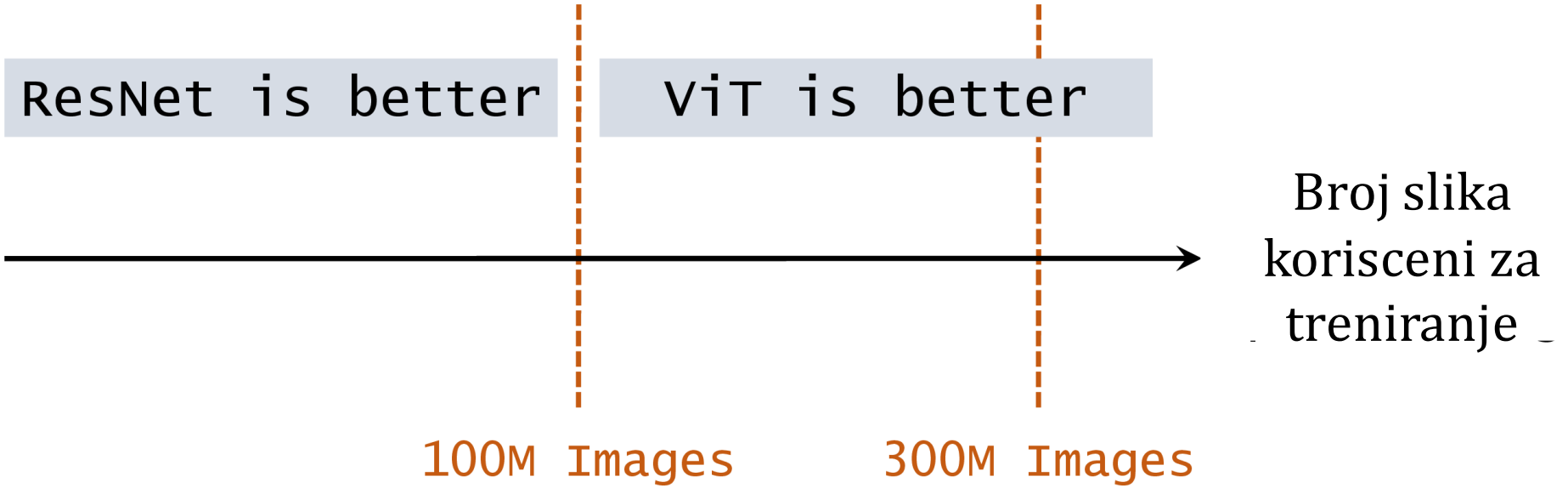

$\mathbf{c}_0$
$\mathbf{c}_1$
$\mathbf{c}_2$
$\mathbf{c}_3$
$\cdots$
$\mathbf{c}_n$
Transformer Encoder Network
$\mathbf{z}_0$
$\mathbf{z}_1$
$\mathbf{z}_2$
$\mathbf{z}_3$
$\cdots$
$\mathbf{z}_n$
Embed
Dense
Dense
Dense
$\cdots$
Dense
[CLS]
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\cdots$
$\mathbf{x}_n$


Softmax
Classifier
p
$\mathbf{c}_0$
Transformer Encoder Network
$\mathbf{z}_0$
$\mathbf{z}_1$
$\mathbf{z}_2$
$\mathbf{z}_3$
$\mathbf{z}_n$
Embed
Dense
Dense
Dense
Dense
[CLS]
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_n$


Softmax
Classifier
c_0
p


Confidence
0.4
0.2
0.12
0.06
0.05
0.04
0.06
0.07
bird
car
cat
dog
fox
jet
snake
tiger
Classes

Tacnost u klasifikaciji slika



Swin Transformer

Pogled na arhitekturu

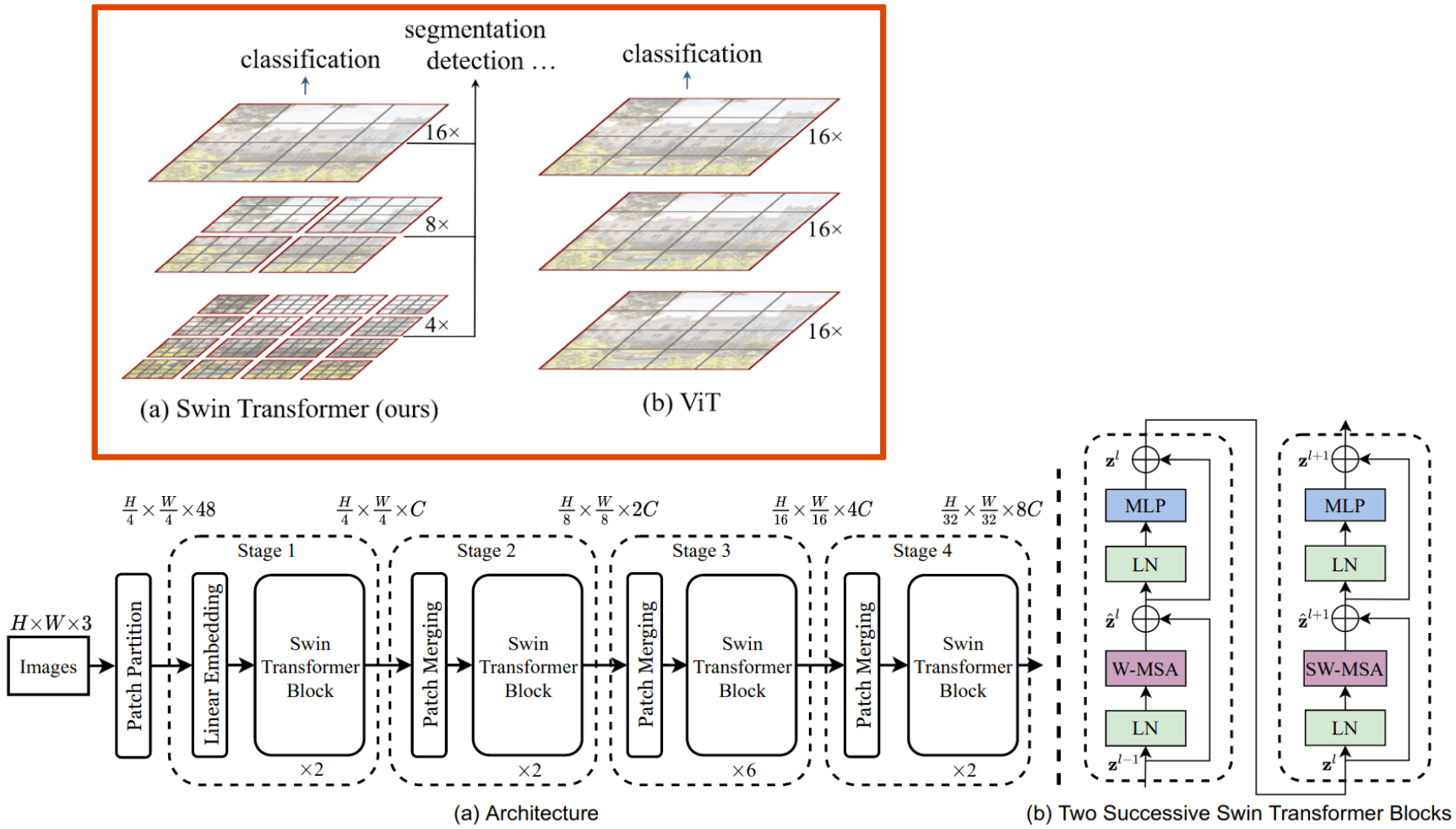
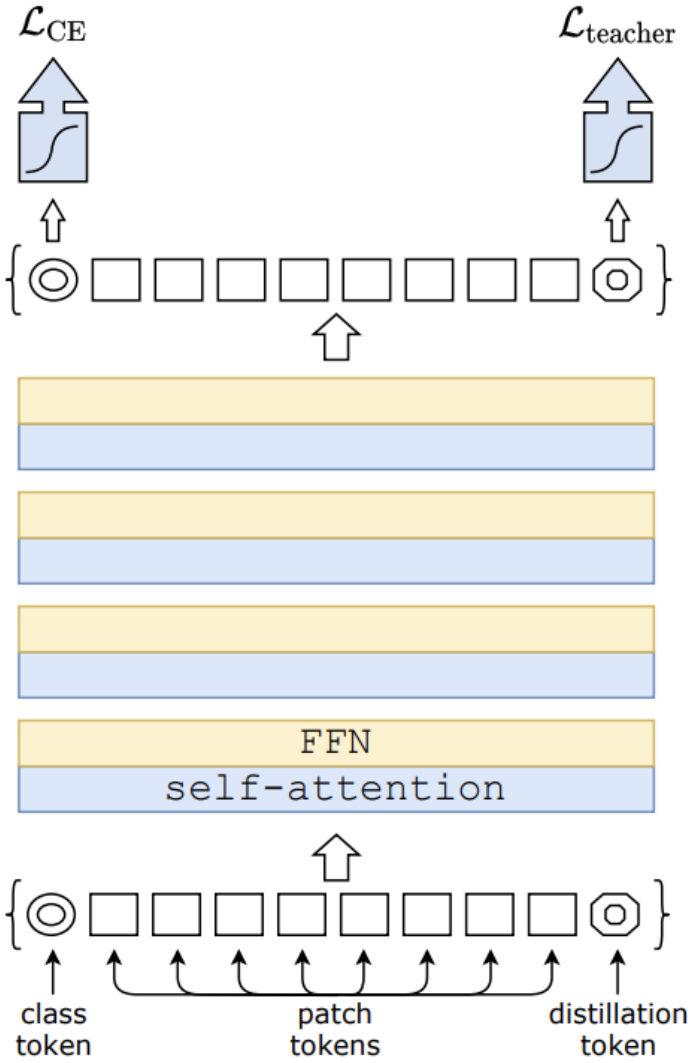


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

DeiT (Data efficient image Transformer)

Pogled na arhitekturu



Rezultati Transformera

Rezultati modernih modela nad poznatim skupom podataka ImageNet-1K

(a) Regular ImageNet-1K trained models					
method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [48]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [48]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [48]	224^2	84M	16.0G	334.7	82.9
EffNet-B3 [58]	300^2	12M	1.8G	732.1	81.6
EffNet-B4 [58]	380^2	19M	4.2G	349.4	82.9
EffNet-B5 [58]	456^2	30M	9.9G	169.1	83.6
EffNet-B6 [58]	528^2	43M	19.0G	96.9	84.0
EffNet-B7 [58]	600^2	66M	37.0G	55.1	84.3
ViT-B/16 [20]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [20]	384^2	307M	190.7G	27.3	76.5
DeiT-S [63]	224^2	22M	4.6G	940.4	79.8
DeiT-B [63]	224^2	86M	17.5G	292.3	81.8
DeiT-B [63]	384^2	86M	55.4G	85.9	83.1
Swin-T	224^2	29M	4.5G	755.2	81.3
Swin-S	224^2	50M	8.7G	436.9	83.0
Swin-B	224^2	88M	15.4G	278.1	83.5
Swin-B	384^2	88M	47.0G	84.7	84.5

Moji rezultati nad skupom podataka Chiro10

Model	Velicina slike	"Chrio10" top-1 acc.
Standard ViT	**512x512**	89.17%
Swin-**T**iny	**224x224**	54.33%
Deit-**S**mall	**224x224**	91.88%

Zaključak

- Vision Transformeri su bili izuzetno popularna tema u poslednjih ~1-2 godine!
- Vrlo različita arhitektura u odnosu na tradicionalne CNN-ove.
- Primena na skoro sve zadatke: klasifikacija, detekcija, segmentacija, itd.
- Glavna prednost je brzina: Množenje matrica je prilagođenije za hardver od konvolucije, pa ViT sa istim FLOPs kao CNN-ovi mogu trenirati i raditi mnogo brže.
- **Moj zaključak**: Vision transformeri predstavljaju evoluciju, a ne revoluciju. Još uvek možemo u osnovi rešavati iste probleme kao sa CNN-ovima.