Adaptivni diskretni sistemi i neuralne mreže

Konvolucione neuralne mreže

Ljubiša Stanković, Miloš Daković, Miloš Brajović

Univerzitet Crne Gore
Elektrotehnički fakultet

Prezentacija 1

Problem dimenzionalnosti pri klasifikaciji

- Kada želimo da klasifikujemo realnu sekvencu podataka, broj uzoraka po sekvenci može biti veoma velik.
- Ovo je posebno izraženo kod slika kao ulaznih sekvenci.
- Na primjer, ako želimo da piksele slike primijenimo na ulaz standardne neuralne mreže, svaki piksel se tretira kao element jedne sekvence za treniranje.
- Svaki piksel mora biti povezan sa svakim neuronom ulaznog sloja.
- **Slika skromne rezolucije** u sivim tonovima sadrži desetine hiljada piksela.
- Ako skriveni sloj ima, recimo, 1024 neurona, potrebno je trenirati **desetine miliona** težinskih koeficijenata.
- Takva mreža bi bila toliko velika da je gotovo nemoguće trenirati je.

Povezanost lokalnih uzoraka i karakteristike

- U većini slučajeva, grupe susjednih uzoraka ili piksela su međusobno povezane.
- Te grupe formiraju **objekte (karakteristike)** u analiziranim signalima i slikama.
- Umjesto analize pojedinačnih uzoraka, možemo tražiti **lokalizovane karakteristike** u signalu ili slici.
- Fokusiranjem na karakteristike, a ne na pojedinačne uzorke, rješava se problem **promjene pozicije**.
- Ako se određena karakteristika (feature) pomjeri, pristup po uzorcima vidi kompletnu promjenu.
- Pristup po karakteristikama prepoznaje isti oblik bilo gdje u signalu ili slici.

Konvolucione neuralne mreže i traženje karakteristika

- **Konvolucione neuralne mreže** vrše pretragu karakteristika u analiziranom signalu ili slici.
- Zbog toga su **invarijantne** na promjenu pozicije oblika.
- Prozor koji se koristi u konvoluciji, **konvolucioni filtar ili jezgro**, sposoban je da prepozna karakteristiku koja odgovara njegovom obliku.
- Pomjeranjem konvolucionog prozora/filtra tražimo poklapanja širom signala ili slike.
- Ovaj proces je sličan korišćenju uveličavajućeg stakla koje prelazi preko slike u potrazi za specifičnim oblicima.

Zašto koristiti konvoluciju?

- Postavlja se pitanje: **kako znamo** da je konvolucija odgovarajuća operacija za detekciju karakteristika u signalu?
- Prema **teoriji prilagođenih filtara**, konvolucija signala sa karakteristikom koju tražimo daje najbolji odgovor na pitanje o njenom prisustvu.
- Prisjetimo se da je izlaz usklađenog filtera dat izrazom:

$$\begin{aligned} y(n) &= x(n) * w(-n) = \sum_m x(m)w(m-n) \\ &= \sum_m x(n+m)w(m) = x(n) *_c w(n), \end{aligned}$$

gdje $w(n)$ predstavlja karakteristiku koju tražimo u signalu $x(n)$.

Detekcija karakteristika i pomjeranje

- Najbolja funkcija za detekciju karakteristike $w(n)$ u signalu $x(n)$ je **konvolucija** signala sa $w(-n)$.
- **Konvoluciona detekcija** daje rezultat koji je nezavisan od pozicije karakteristike.
- Izračunava se tako što se prozor/filtar $w(n)$ pomjera duž signala i množi sa signalom $x(n)$.
- Isto važi i za slike: **dvodimenzionalni filter** $w(m,n)$ se pomjera duž slike u oba prostorna smjera.
- Računa se konvolucija sa $w(-m,-n)$.

Konvencija i terminologija kod konvolucionih neuralnih mreža

- U definiciji prilagođenog filtra, konvolucija $x(n) * w(-n)$ više liči na **kros-korelaciju** nego na klasičnu konvoluciju $x(n) * w(n)$.
- Ipak, mreže se nazivaju **konvolucione neuralne mreže (CNN)**, a ne korelacione neuralne mreže.
- U literaturi se koriste razne notacije za obrtanje signala, npr. $rot180^0(\mathbf{w}) * \mathbf{x}$ ili $\text{conv}(\text{reverse}(\mathbf{w}), \mathbf{x})$.
- Mi ćemo koristiti jednostavnu notaciju $\mathbf{w} *_c \mathbf{x}$, što znači da je prvi signal u konvoluciji obrnut.

Uloga konvolucionih slojeva kod konvolucionih neuralnih mreža

- Konvolucione neuralne mreže (engl. *Convolutional Neural Networks* – **CNN**) su neuralne mreže koje koriste **konvolucione slojeve**.
- Svaki konvolucioni sloj sadrži skup **konvolucionih filtara**.
- Ti filtri transformišu ulazne signale ili slike u novi skup signala ili slika koristeći konvoluciju.
- U CNN-ovma obično tražimo više različitih karakteristika u signalu ili slici.
- Zato se koristi više konvolucionih filtera, gdje svaki prepoznaje različitu karakteristiku.
- Učeći različite oblike iz prostora karakteristika, CNN omogućava jednostavnu, robustnu i efikasnu analizu i klasifikaciju signala i slika.

Propagacija unaprijed

- Algoritam konvolucione neuralne mreže (CNN) započinje izračunavanjem svih signala od ulaza do izlaza, uz pretpostavljene (inicijalizovane) ili prethodno naučene težine konvolucionih filtara — **propagacija unaprijed**.
- Kasnije će biti analizirano ažuriranje težina.
- **Ulaz:** posmatramo signal $\mathbf{x}$ dužine N uzoraka:

$$\mathbf{x} = [x(0),\ x(1),\ \ldots,\ x(N-1)]^T$$

- Alternativno, ulaz može biti slika dimenzija $N \times N$.
- Zadatak CNN mreže je da klasifikuje ulazne signale (ili slike) u nekoliko grupa (klastera).

Propagacija unaprijed: konvolucija

- Na signal $\mathbf{x}$ se primjenjuje **konvolucioni filtar** dužine M — ili filtar dimenzija $M \times M$ u slučaju slike.
- Tipične vrijednosti su $M = 3$ ili $M = 5$.
- Pošto tragamo za više karakteristika (*features*) u signalu $x(n)$, koristi se K takvih filtara.
- Težine u prvom konvolucionom sloju su:

$$\mathbf{w}_k^1 = [w_k^1(0),\ w_k^1(1),\ \ldots,\ w_k^1(M-1)]^T, \quad k = 1, 2, \ldots, K$$

- Pretpostavljamo da se ulaz ne dopunjava nulama (nema *zero-padding*-a).

Propagacija unaprijed: rezultat konvolucije

- Izlazni signali dobijeni konvolucijom:

$$\mathbf{y}_k^1 = \mathbf{w}_k^1 *_c \mathbf{x}$$

- Ovdje $*_c$ označava konvoluciju sa obrnutim filtrom — tj. cross-korelaciju.
- Za $M = 3$, izraženo po komponentama:

$$y_k^1(n) = w_k^1(0)x(n) + w_k^1(1)x(n+1) + w_k^1(2)x(n+2)$$

$$= \sum_{m=0}^{M-1} w_k^1(m)x(n+m)$$

- Ako nije korišćen *zero-padding*, dimenzija $\mathbf{y}_k^1$ je $(N-M+1) \times 1$

Dimenzije izlaza i broj parametara

- Posljednji uzorak u $y_k^1(n)$ se dobija kada je $n+m=N-1$, odnosno $n=N-M$.
- Dakle, posljednji element je $y_k^1(N-M)$, a za $M=3$ to je $y_k^1(N-3)$.
- Računa se ukupno K izlaznih signala $\mathbf{y}_k^1$, za $k=1,2,\ldots,K$.
- Ukupan broj izlaznih uzoraka u prvom konvolucionom sloju je $K(N-M+1)$.
- Broj težina filtara $w_k^1(n)$ je MK i znatno je manji nego kod potpuno povezanih mreža, gdje bi broj veza bio NK.

Generalizacija na slike

- Ako se posmatra slika, izlazna slika ima dimenzije $(N-M+1) \times (N-M+1)$.
- Ima ukupno K takvih izlaznih slika (po jedan izlaz po filtru).
- Ukupan broj težina konvolucionih filtara u tom slučaju je KM^2.

Primjer: CNN kao poseban slučaj standardne mreže (1)

Primjer 1

Povezanost sa standardnom neuralnom mrežom. Odnos ulaz–izlaz u konvolucionim mrežama može proizvesti standardnu mrežu kao **poseban slučaj**. U standardnoj mreži, veza je:

$$y(n) = \sum_{k=1}^{N} w_k x_k(n)$$

gdje $x_k(n)$ predstavlja ulaz u neuron k u trenutku n.
Kod CNN-a, jedan signal $x(n)$ se tretira kao ulaz sa N uzoraka koji simultano dolaze na N ulaznih neurona. Ovi podaci se povezuju sa K izlaznih neurona, pa je veza:

$$y_k = \sum_{m=0}^{N-1} w_k^1(m)x(m) = w_k^1(0)x(0) + \cdots + w_k^1(N-1)x(N-1)$$

Primjer: CNN kao poseban slučaj standardne mreže (2)

Primjer 1

U novoj notaciji, izostavlja se vremenski indeks trenažnog seta, a $x(n)$ se primjenjuje kao cjelina. Indeks uzorka n odgovara indeksu ulaznog neurona. Indeks k označava izlazni neuron, m ulazni, a oznaka 1 odnosi se na sloj.
Formula konvolucije za $M = N$ postaje:

$$y_k^1(n) = w_k^1(0)x(n) + w_k^1(1)x(n+1) + \cdots + w_k^1(N-1)x(n+N-1)$$

Bez dopune nulama, izražava se samo za $n = 0$:

$$y_k^1 = y_k^1(0) = w_k^1(0)x(0) + \cdots + w_k^1(N-1)x(N-1)$$

što je isto kao y_k, dakle standardna mreža je **poseban slučaj CNN-a** kada je $M = N$.

Primjer: CNN kao poseban slučaj standardne mreže (3)

Primjer 1

Napomena: svi rezultati za konvolucione slojeve važe i za potpuno povezane slojeve kada se koristi $M = N$ i $n = 0$.

Kod standardne mreže: $x(n)$ za $n = 0, \ldots, N-1$ ide na N ulaznih neurona povezanih sa K izlaznih neurona y_k.

Broj težina je NK.

Kod CNN: signal $x(n)$ se pomjera kroz M uzastopnih uzoraka i množi istim težinama $w_k^1(0), \ldots, w_k^1(M-1)$, za sve n, fiksno po k.

Primjer: CNN kao poseban slučaj standardne mreže (4)

Primjer 1

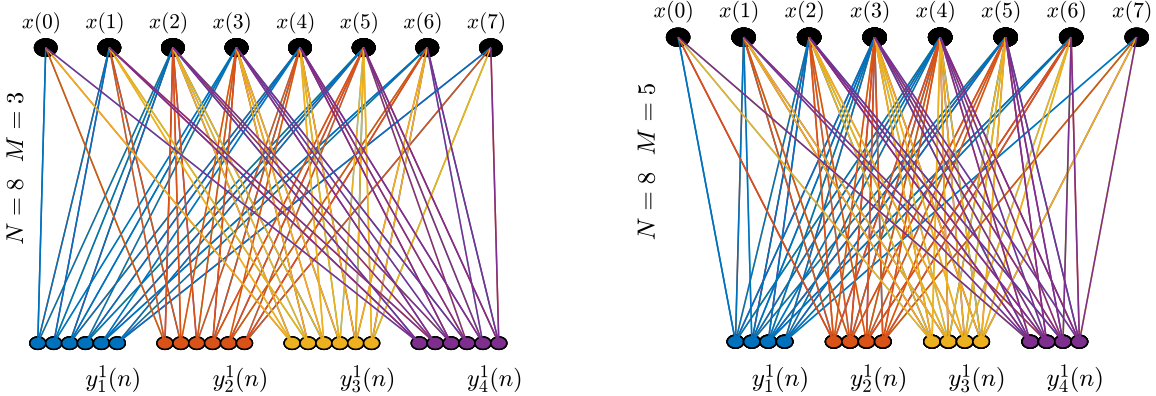
U vektorskoj notaciji imamo:

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix} \qquad \mathbf{W}^1 = \begin{bmatrix} w_{k,1} \\ w_{k,2} \\ \vdots \\ w_{k,N} \end{bmatrix} = \begin{bmatrix} w_k^1(0) \\ w_k^1(1) \\ \vdots \\ w_k^1(N-1) \end{bmatrix}$$

za $w_k(n) = w_{k,n}$ i

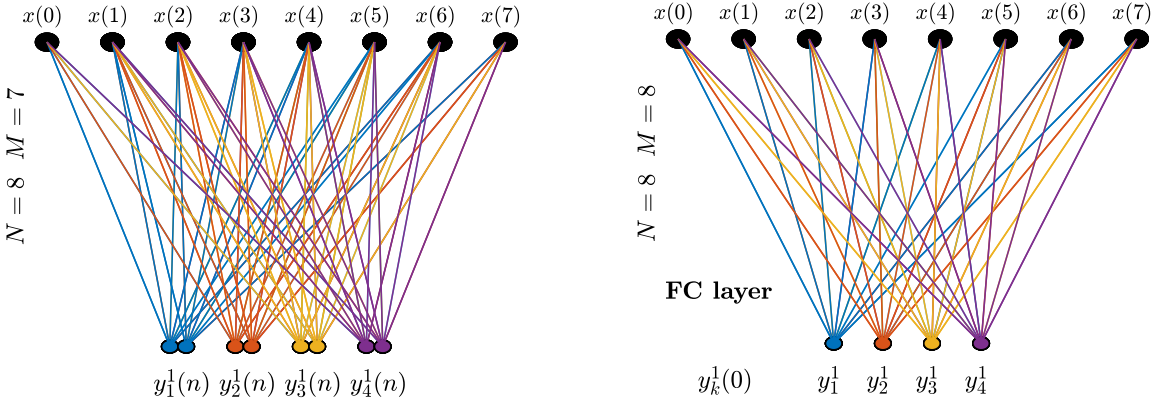
$$y_k^1 = (\mathbf{W}^1)^T \mathbf{X}$$

Ilustracija: CNN sloj za $M = 3$ i $M = 5$



Slika: CNN sloj sa $N = 8$ i $M = 3$ (lijevo), $M = 5$ (desno), sa $K = 4$ izlaza.

Ilustracija: CNN sloj za $M = 7$ i $M = 8$



Slika: Za $M = 7$ i $M = 8$ (potpuno povezani sloj). CNN sa $M = N$ postaje FC sloj.

Bias u konvolucionim slojevima

- **Bias:** kao i u standardnim slojevima, u konvolucionom sloju se dodaje konstanta (bias) na izlazni signal:

$$y_k^1(n) = \sum_{m=0}^{M-1} w_k^1(m) x(n+m) + b_k^1$$

- U vektorskom obliku:

$$\mathbf{y}_k^1 = \mathbf{w}_k^1 *_c \mathbf{x} + b_k^1$$

- Time se broj parametara u svakom konvolucionom izrazu povećava za 1.
- **Bez obzira na veličinu signala (ili slike), broj težina u CNN-u zavisi samo od veličine filtra.**
- Ukupno: $K(M+1)$ parametara za signale; za slike $K(M^2+1)$.

Zero-padding i očuvanje dimenzije

- **Zero-padding:** nekad želimo da rezultat konvolucije ima **istu dužinu** kao ulazni signal (ili slika), tj. N umjesto $(N-M+1)$.
- To postižemo dopunjavanjem ulaza nulama (npr. $x(-1)=0$, $x(N)=0$ za $M=3$).
- Tada konvolucija izgleda kao:

$$y_k^1(n) = w_k^1(0)x(n-1) + w_k^1(1)x(n) + w_k^1(2)x(n+1)$$

- Broj uzoraka izlaza sada je isti kao kod ulaza: $y(0)$ do $y(N-1)$.

Opšti slučaj zero-paddinga

- Za filtar neparne dužine M, da bi rezultat imao istu dimenziju kao ulaz, signal treba proširiti sa $(M-1)/2$ nula prije $n=0$ i isto toliko nakon $n=N-1$.
- Ovakav pristup omogućava da se **očuva dužina signala** i koristi u arhitekturama gdje je potrebna konzistentnost dimenzija slojeva.

Nelinearna aktivaciona funkcija

- **Nelinearna aktivacija:** pošto je konvolucija linearna, a signali/slike nisu, nakon konvolucionog sloja primjenjujemo nelinearnu funkciju.
- Najčešće se koristi **ReLU (Rectified Linear Unit)**:

$$f(x) = \max\{0, x\}$$

- Prednosti ReLU funkcije:
  - Ne zasićuje se za pozitivne vrijednosti i održava gradijent.
  - Jednostavna za računanje (bez eksponencijalnih funkcija).
  - Brže konvergira od *sigmoid* i *tanh*.

ReLU i sparsifikacija

- ReLU ne aktivira sve neurone istovremeno — **negativni ulazi se deaktiviraju**".
- To uvodi **sparsifikaciju** izlaza, što doprinosi efikasnosti.
- Izlaz konvolucionog sloja sa aktivacijom:

$$f(\mathbf{y}_k^1) = f(\mathbf{w}_k^1 *_c \mathbf{x} + b_k^1) = f(\mathbf{w}_k^1, \mathbf{x})$$

- Za $M = 3$:

$$f(y_k^1(n)) = f\left(w_k^1(0)x(n) + w_k^1(1)x(n+1) + w_k^1(2)x(n+2) + b_k^1\right)$$

Primjer 3: ReLU aktivacija (1)

Primjer 2

Posmatrajmo signal $\mathbf{y}_k = \mathbf{w}_k^1 *_c \mathbf{x} + b_k$ na $K = 3$ kanala, $k = 1, 2, 3$:

$$\mathbf{y}_1 = \begin{bmatrix} 0.35 & 0.49 & -0.65 & -0.65 & -0.69 & 0.48 \end{bmatrix}^T$$

$$\mathbf{y}_2 = \begin{bmatrix} -0.05 & -0.06 & -0.28 & -0.21 & 0.13 & 0.37 \end{bmatrix}^T$$

$$\mathbf{y}_3 = \begin{bmatrix} 0.48 & 0.50 & -0.77 & -1.66 & -0.76 & 0.71 \end{bmatrix}^T$$

Izlaz ReLU aktivacione funkcije:

$$f(\mathbf{y}_1) = \begin{bmatrix} 0.35 & 0.49 & \mathbf{0.00} & \mathbf{0.00} & \mathbf{0.00} & 0.48 \end{bmatrix}^T$$

$$f(\mathbf{y}_2) = \begin{bmatrix} \mathbf{0.00} & \mathbf{0.00} & \mathbf{0.00} & \mathbf{0.00} & 0.13 & 0.37 \end{bmatrix}^T$$

Primjer 3: ReLU aktivacija (2)

Primjer 2

$$f(\mathbf{y}_3) = \begin{bmatrix} 0.48 & 0.50 & \mathbf{0.00} & \mathbf{0.00} & \mathbf{0.00} & 0.71 \end{bmatrix}^T$$

Matrica indikatora deaktivacije neurona:

$$\mathbf{M}^{ReLU} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Problem ReLU aktivacije i Leaky ReLU

- Glavni problem kod **ReLU** funkcije: kada neuroni ostanu ,,zarobljeni" u negativnim vrijednostima.
- Ako neuron stalno ima negativan izlaz, $f(x) = 0$, on ne ažurira težine – **"umrli neuron" (dying ReLU)**.
- Rješenje: **Leaky ReLU** – negativne vrijednosti se skaliraju:

$$f(y_k(n)) = \begin{cases} y_k(n), & y_k(n) \geq 0 \\ 0.01 \cdot y_k(n), & y_k(n) < 0 \end{cases}$$

Konvolucioni korak – **stride**

- Prilikom računanja konvolucije, filtar se pomjera po 1 korak: $n \to n+1$.
- Ako se želi dodatno sažimanje izlaza, koristi se **stride** – preskakanje *instanata*.
- Npr. stride = 2 znači da se konvolucija računa na svakom drugom uzorku.
- Stride = 4 $\to$ računa se $y_k(n)$ samo na svaka 4 uzorka ulaza $x(n)$.
- Stride je vrsta **down-samplinga**.

Primjer

Primjer 3

Posmatrajmo izlaze iz ReLU funkcije iz Primjera 2:

$$f(\mathbf{y}_1) = \begin{bmatrix} \mathbf{0.35} & 0.49 & 0.00 & \mathbf{0.00} & 0.00 & 0.48 \end{bmatrix}^T$$

$$f(\mathbf{y}_2) = \begin{bmatrix} \mathbf{0.00} & 0.00 & 0.00 & \mathbf{0.00} & 0.13 & 0.37 \end{bmatrix}^T$$

$$f(\mathbf{y}_3) = \begin{bmatrix} \mathbf{0.48} & 0.50 & 0.00 & \mathbf{0.00} & 0.00 & 0.71 \end{bmatrix}^T$$

Sa stride $= 3$ dobijamo:

$$\text{Stride}_3\{f(\mathbf{y}_1)\} = \begin{bmatrix} 0.35 & 0.00 \end{bmatrix}^T, \quad \text{Stride}_3\{f(\mathbf{y}_2)\} = \begin{bmatrix} 0.00 & 0.00 \end{bmatrix}^T$$

$$\text{Stride}_3\{f(\mathbf{y}_3)\} = \begin{bmatrix} 0.48 & 0.00 \end{bmatrix}^T$$

Primjer: matrica za upsampling poslije stride

Primjer 3

Matrica za vraćanje stride-ovanih podataka na originalnu dužinu:

$$\mathbf{M}^{Stride_3} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}^T$$

Pooling: princip i efekti

- Osim stride, koristi se i **pooling** za sažimanje podataka.
- Najčešće: **max-pooling** — signal se dijeli na nepokrivajuće segmente od P uzoraka i bira se maksimum.
- Kod slike: dijeljenje na kvadratne regione $P \times P$.
- Izlaz:
$$o_k^1(n) = \max\{f(y_k(n)), f(y_k(n+1)), \ldots, f(y_k(n+P-1))\}$$
- Vektorski zapis: $\mathbf{o}_k^1 = F_1(\mathbf{w}_k^1, \mathbf{x})$
- Prednosti: manja dimenzija, manje parametara, parcijalna **translaciona invarijantnost**.

Primjer: Max-pooling $P = 3$

Primjer 4

Max-pooling ($P = 3$) na $f(\mathbf{y}_k)$ iz Primjera **??** daje:

$$\mathbf{o}^1 = \begin{bmatrix} \max\{0.35, 0.49, 0.00\} & \max\{0.00, 0.00, 0.48\} \\ \max\{0.00, 0.00, 0.00\} & \max\{0.00, 0.13, 0.37\} \\ \max\{0.48, 0.50, 0.00\} & \max\{0.00, 0.00, 0.71\} \end{bmatrix}^T = \begin{bmatrix} 0.49 & 0.48 \\ 0.00 & 0.37 \\ 0.50 & 0.71 \end{bmatrix}^T$$

Matrica za vraćanje iz max-pooled na originalnu dužinu (upsampling):

$$\mathbf{M}^{MP} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}^T$$

Napomena: max-pooling sa $P = 3$ smanjuje dužinu sa 6 na 2 — kao i stride 3 — ali odabrane pozicije zavise od samog signala.

Proces **flattening-a** signala

- Nakon pooling-a ili stride-a, vektori izlaza se **spajaju** u jedan vektor:

$$o_F^1((k-1)(N-M+1)+n) = o_k^1(n)$$

  za $k = 1, 2, \ldots, K$ i $n = 0, 1, \ldots, N-M$.
- Ako nema max-poolinga: veličina $K(N-M+1)$.
- Ako postoji max-pooling sa faktorom P: veličina $K(N-M+1)/P$.
- Kod slika: 2D objekti se linearizuju u jedan vektor.
- Ovaj proces nazivamo **flattening**.

Primjer: prvi konvolucioni sloj CNN-a

Primjer 5

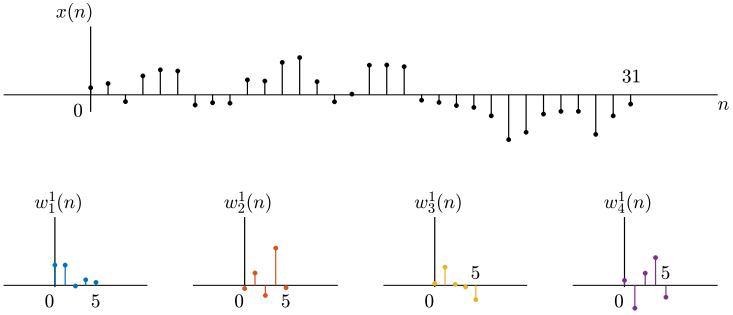
Flattenovana forma $\mathbf{o}^1$ iz primjera 32:

$$\mathbf{o}_F^1 = \begin{bmatrix} 0.49 & 0.48 & 0.00 & 0.37 & 0.50 & 0.71 \end{bmatrix}^T$$

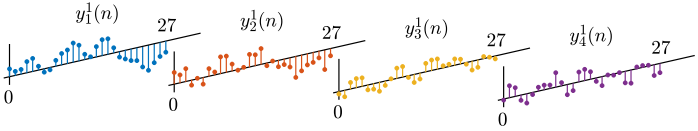
Prvi sloj CNN-a:

- Ulazni signal: $N = 32$ uzoraka
- Četiri konvoluciona filtra ($K = 4$) dužine $M = 5$
- ReLU aktivacija: $\max\{0, y_k^1 + b_k^1\}$
- Max-pooling sa faktorom $P = 2$
- Filtri inicijalizovani Gaussovim raspodjelama

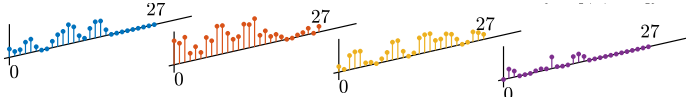
Ilustracija: prvi konvolucioni sloj (1/2)



$$y_k^1(n) = x(n) * w_k^1(-n) = x(n)w_k^1(0) + x(n+1)w_k^1(1) + x(n+2)w_k^1(2) + x(n+3)w_k^1(3) + x(n+4)w_k^1(4)$$

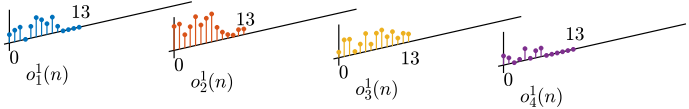


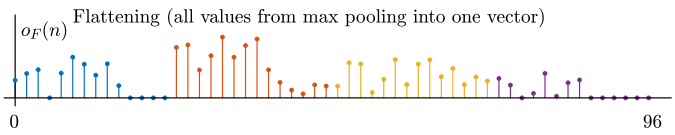
Ilustracija: prvi konvolucioni sloj (2/2)



$[b_1^1,\ b_2^1,\ b_3^1,\ b_4^1] = [0.05,\ 0.8,\ 0.5,\ -0.15]$

Max-pooling by factor of $P = 2$





Input to the fully connected (FC) neural network

Objašnjenje: prvi konvolucioni sloj

- Ulazni signal sadrži $N = 32$ uzoraka.
- Četiri konvoluciona filtra ($K = 4$) dužine $M = 5$ uzoraka.
- Na izlazu konvolucije primijenjena je ReLU nelinearna aktivacija $\max\{0, y_k^1 + b_k^1\}$.
- Max-pooling sa faktorom $P = 2$: signal je podijeljen u segmente od dva uzorka, a uzima se maksimalna vrijednost iz svakog segmenta.
- Izlaz iz max-poolinga se koristi kao ulaz za naredni konvolucioni sloj ili se flatten-uje i predaje potpuno povezanom (FC) sloju neuronske mreže.
- Težine filtara su inicijalizovane Gaussovim slučajnim brojevima.

Ponavljanje konvolucija

- Prije flattenovanja, **konvolucioni koraci** se mogu ponavljati.
- Novi skupovi filtara pronalaze **hijerarhijske karakteristike (*features*)**.
- Redoslijed koraka: konvolucija $\rightarrow$ aktivacija $\rightarrow$ pooling, pa opet konvolucija itd.

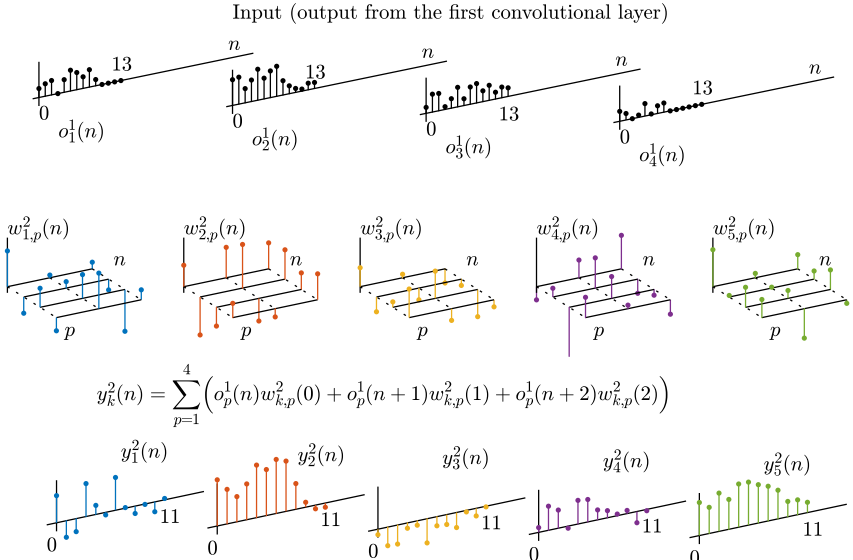
Primjer: drugi konvolucioni sloj CNN-a

Primjer 6

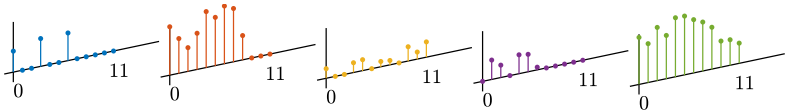
Izlazi $o_1^1(n), \ldots, o_4^1(n)$ iz prvog sloja ulaze u drugi sloj.

- Pet konvolucionih filtara ($K = 5$) dužine $M = 3$
- Računaju se izlazi $y_1^2(n), \ldots, y_5^2(n)$
- ReLU aktivacija: $\max\{0, y_k^2(n) + b_k^2\}$
- Max-pooling sa faktorom $P = 2$
- Flattenovan izlaz: $o_F^2(n)$

Ilustracija: drugi konvolucioni sloj (1/2)

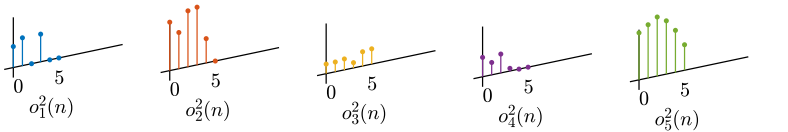


Ilustracija: drugi konvolucioni sloj (2/2)

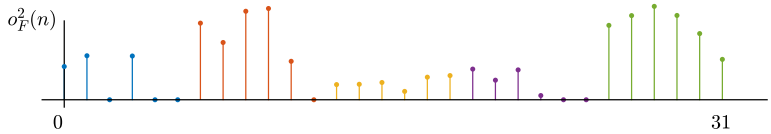


$$[b_1^2,\ b_2^2,\ b_3^2,\ b_4^2,\ b_5^2] = [-0.55,\ -0.3,\ 0.8,\ -0.45,\ 0.4]$$

Max pooling by factor of $P = 2$



Flattening (all values from max pooling into one vector), $o_F^2((k-1)M+m) = o_k^1(m)$



Input to fully connected (FC) neural network

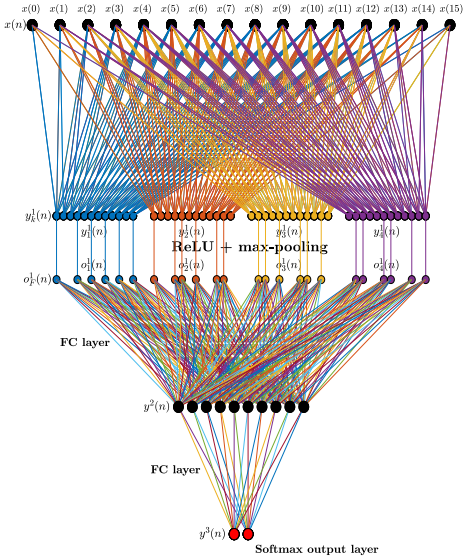
Objašnjenje: drugi konvolucioni sloj

- Izlazni signali $o_1^1(n)$, $o_2^1(n)$, $o_3^1(n)$ i $o_4^1(n)$ iz prvog sloja koriste se kao ulaz u drugi konvolucioni sloj.
- Svaki od ovih signala se procesuira sa pet konvolucionih filtara ($K = 5$) dužine $M = 3$.
- Dobijaju se izlazi $y_k^2(n)$ ($k = 1, 2, 3, 4, 5$).
- Primijenjena je ReLU aktivacija $\max\{0, y_k^2 + b_k^2\}$.
- Nakon toga, primijenjen je max-pooling sa faktorom $P = 2$.
- Izlaz max-poolinga se flatten-uje u vektor $o_F^2(n)$ kao priprema za potpuno povezani sloj.
- Težine filtara su inicijalizovane Gaussovim slučajnim brojevima.

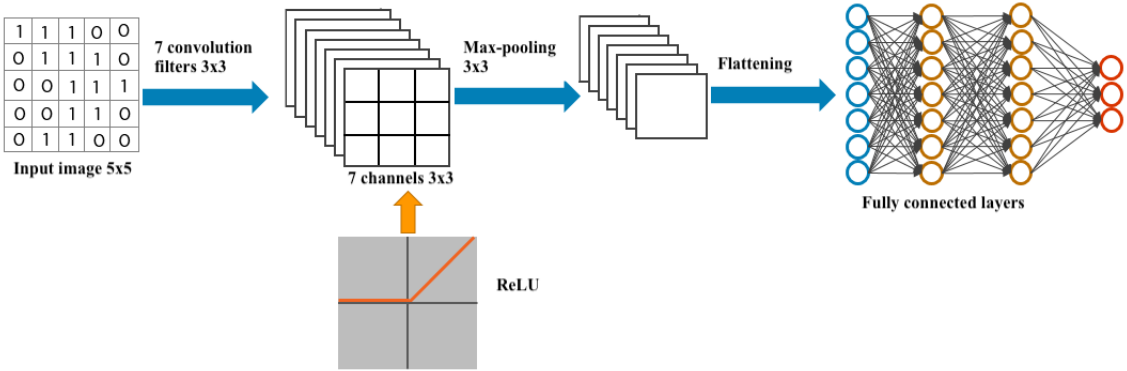
Potpuno povezani slojevi (FC)

- Izlazi iz prethodnih konvolucionih koraka, nakon flattening-a, povezuju se na standardnu neuronsku mrežu sa potpuno povezanim neuronima.
- Neuroni u ovom sloju imaju punu povezanost sa svim neuronima u prethodnom i narednom sloju, kao u regularnim feed-forward mrežama.
- Zbog toga se izlaz iz ovog sloja računa kao i u običnim neuronskim mrežama.
- FC sloj pomaže u preslikavanju reprezentacije između ulaza i izlaza.
- FC slojevi mogu imati tradicionalnu višeslojnu strukturu, kao što je ranije opisano.
- Na izlazu se najčešće koristi softmax sloj kada se CNN koristi za klasifikaciju signala.

Ilustracija: 1D CNN sa jednim konvolucionim dva FC sloja



Ilustracija: povezivanje na potpuno povezane slojeve



Dvodimenzioni CNN sa jednim konvolucionim slojem i dva FC sloja.

Primjer: jedan CNN sloj sa 32 uzorka

Primjer 7

- Ulazni signal $x(n)$ ima $N = 32$ uzorka.
- Signal se primjenjuje na ulaz jednodimenzionalnog konvolucionog sloja CNN-a (pogledati prethodne slike).
- Koriste se $K = 4$ konvoluciona filtra: $w_1^1(n)$, $w_2^1(n)$, $w_3^1(n)$, $w_4^1(n)$ dužine $M = 3$.
- Izlazi konvolucije su: $y_1^1(n)$, $y_2^1(n)$, $y_3^1(n)$, $y_4^1(n)$.
- Na izlaz se primjenjuje ReLU: $\max\{0, y_k^1(n) + b_k\}$, $k = 1, \ldots, 4$.
- Slijedi max-pooling sa faktorom $P = 2$, čime se dobijaju: $o_1^1(n)$, $o_2^1(n)$, $o_3^1(n)$, $o_4^1(n)$.
- Konačno, izlazi se spajaju u jedan vektor: $o_F^1(n)$.

Primjer: arhitektura AlexNet

Primjer 8

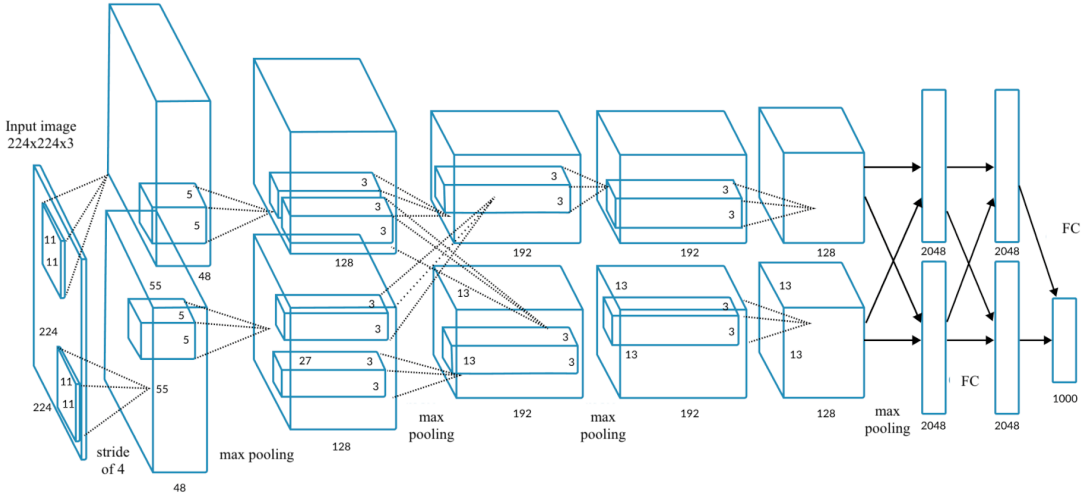
- CNN treniran nad 1.2 miliona slika (ImageNet LSVRC-2010), klasifikovanih u 1000 klasa.
- Mreža ima:
  - 60 miliona parametara,
  - 650.000 neurona,
  - 5 konvolucionih slojeva (neki sa max-pooling),
  - 3 potpuno povezana (FC) sloja,
  - završni softmax sa 1000 klasa.
- Konvolucioni slojevi:
  - CONV1: $96 \times 11 \times 11 \times 3$, stride 4
  - CONV2: $256 \times 5 \times 5 \times 48$
  - CONV3: $384 \times 3 \times 3 \times 256$

Primjer: arhitektura AlexNet

Primjer 9

- Konvolucioni slojevi (nastavak):
  - CONV4: $384 \times 3 \times 3 \times 192$
  - CONV5: $256 \times 3 \times 3 \times 192$
- Max-pooling nakon slojeva 1, 2 i 5.
- FC slojevi:
  - FC6: 4096 neurona
  - FC7: 4096 neurona
  - FC8: 1000 neurona (klasifikacija)
- Preuzeto iz: A. Krizhevsky, I. Sutskever, G. Hinton (2017), *Communications of the ACM*, 60(6): 84–90.

Ilustracija: arhitektura AlexNet



AlexNet

AlexNet mreža (2012) klasifikuje ulazne slike dimenzija $224 \times 224 \times 3$ u 1000 klasa. Sastoji se od:

- 5 konvolucionih slojeva sa max-pooling-om,
- 3 potpuno povezana sloja (4096, 4096, 1000),
- ukupno 60 miliona parametara i 650.000 neurona.