Prepoznavanje saobraćajnih znakova pomoću konvolutivnih neuronskih mreža

Seminarski rad u okviru kursa
Računarska inteligencija
Matematički fakultet

Jana Jovičić 215/2015, Jovana Pejkić 435/2016
jana.jovicic755@gmail.com, jov4ana@gmail.com

16. maj 2019.

Sažetak

Sadržaj

1 Uvod

Prepoznavanje i klasifikacija slika je polje u oblasti mašinskog učenja koje se veoma brzo razvija. Na primer, klasifikatori slika se sve više koristite za: zamenu lozinke prepoznavanjem lica, prepoznavanje otisaka prstiju, analizu krvnih slika, identifikaciju geografskih karakteristika iz satelitskih snimaka, analizu aero i satelitskih snimaka, detekciju urbanih područja, detekciju puteva, autonomnu vožnju i otkrivanje prepreka itd.

Klasifikacija je vrsta problema nadgledanog učenja[1] koja podrazumeva predviđanje kategoričke ciljne promenljive. Kategoričkim se smatraju promenljive koje uzimaju konačan broj vrednosti (među kojima nema uređenja). Na primer, prepoznavanje da li je dobijen e-mail *smeće (eng.* spam*)*, reklama ili obična poruka je problem klasifikacije.

Postoje mnogi algoritmi koji se koriste za klasifikaciju slika (npr. SVM[2]), a jedan od najboljih predstavljaju *konvolutivne neuronske mreže (eng.* convolutional neural network, CNN*)*. CNN se može zamisliti kao automatski „ekstraktor" karakteristika slike. On efikasno koristi informacije o susednim pikselima kako bi konvolucijom i agregacijom smanjio sliku, a zatim predvideo kategoriju kojoj slika pripada. Iako su konvolutivne neuronske mreže projektovane tako da prednost postižu u radu sa 2D strukturama, kao što su slike ili ulazi poput *govornog signala (eng.* speech signal*)*, najnovije studije pokazuju da postižu značajne rezultate i sa 3D strukturama.

U ovom radu je predstavljeno rešavanje problema klasifikacije slika saobraćajnih znakova pomoću neuronskih mreža (koje su opisane u poglavlju 2), i to konvolutivnih neuronskih mreža (koje su opisane u poglavlju 3). Cilj ovog rada je da, kroz rešavanje ovog problema, prikaže osnovne osobine konvolutivnih neuronskih mreža kao i njihovu implementaciju. U radu je, u poglavlju 4, najpre opisano nekoliko razlicitih modela mreža, a onda su upoređeni i analizirani ostvareni rezultati.

[1] Nadgledano učenje (eng. supervised learning) je proces učenja funkcije koja preslikava ulaz u odgovarajući izlaz, zasnovan na datim ulzno-izlaznim parovima. Dve osnovne vrste problema nadgledanog učenja su regresija i klasifikacija. Regresija je problem predvidanja neprekidne, a klasifikacija kategoričke ciljne promenljive.

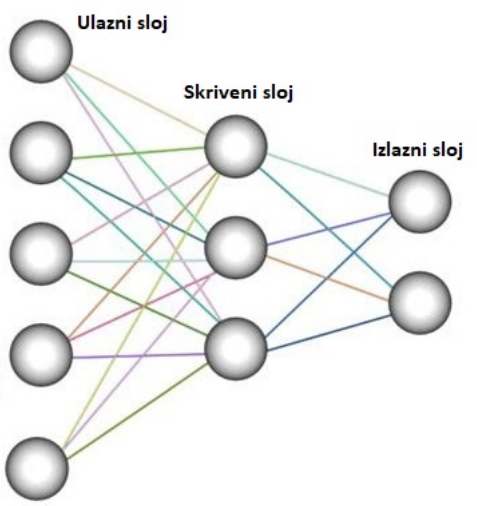
[2] *Podržavajući vektori (eng.* Support vector machines SVM*)* predstavljaju model nadgledanog učenja koji se koristi za klasifikacionu i regresionu analizu.

2 Neuronske mreže

Neuronske mreže predstavljaju skup statističkih modela učenja inspirisana biološkim neuronima, za rešavanje klasifikacionih[3] i regresijskih problema[4]. Njihove primene su mnogobrojne, a neke od njih su: kategorizacija teksta, raspoznavanje i sinteza govora, autonomna voznja, igranje video igara, mašinsko prevođenje prirodnih jezika i slično. Ključna prednost neuronskih mreža je da same mogu da konstruišu nove atribute nad sirovom reprezentacijom podataka[5] i odlično baratanje sa velikim količinama podataka.

2.1 Arhitektura

Osnovnu jedinicu građe neuronske mreže predstavljaju neuroni, koji su međusobno povezani vezama sa težinama koje se podešavaju tokom učenja mreže. Povezani neuroni prosleđuju signale jedni drugima, a organizovani su u slojeve. Najjednostavniji oblik neuronske mreže je **perceptron**, koji sadrži samo jedan ulazni i jedan izlazni sloj. Međutim, kako on služi samo za učenje linearnih modela, a u praksi se javlja potreba da i složeniji modeli mogu da se nauče, osim perceptrona, koriste se višeslojne neuronske mreže. Višeslojna neuronska mreža osim ulaznog i izlaznog sloja, ima jedan ili više skrivenih slojeva (slika 1). Kako se neuronske mreže uglavnom koriste za klasifikaciju uzorka u različite kategorije, ulazni sloj se sastoji od onoliko neurona kolika je dimenzionalnost ulaznog prostora, a broj neurona na izlazu jednak je broju klasa. Samo učenje neuronske mreže je zapravo podešavanje težina sve dok se ne dobije zadovoljavajuća aproksimacija između ulaznih i izlaznih veličina.



Slika 1: Višeslojna neuronska mreža sa jednim skrivenim slojem

[3]Klasifikacioni problem - ako je izlazna promenljiva kategorickog tipa, npr. ,,zdrav"i ,,bolestan".

[4]Regresioni problem - ako je izlazna promenjiva neprekidnog tipa, npr. ,,plata"ili ,,tezina".

[5]Iako se nekad mogu pretpostaviti koji su atributi najinformativniji za predviđanje ciljne promenljive, izbori tih atributa su neretko lošiji od onoga što bi algoritam učenja mogao da detektuje u sirovoj informaciji.

2.2 Razlog uvodjenja konvolutivnih neuronskih mreža

Jednostavni zadaci prepoznavanja mogu se dosta dobro rešiti skupovima podataka malih veličina, npr. desetine hiljada slika. Međutim, objekti u realističnim postavkama pokazuju značajnu raznovrsnost, stoga, da bi bilo moguće naučiti prepoznati ih, potrebno je koristiti mnogo veće skupove za treniranje. Da bi se naučilo o hiljadama objekata iz miliona slika, potreban je model sa velikim kapacitetom učenja. Konvolutivne neuronske mreže, koje su detaljno obrađene u ostatku rada, čine jednu takvu klasu modela. One daju jake i uglavnom ispravne pretpostavke o prirodi slika, a njihov kapacitet se moze kontrolisati variranjem njihove dubine i širine. Tako, u poređenju sa standardnim neuronskim mrežama (*mrežama sa propagacijom unapred (eng.* feedforward neural network*)*) sa slojevima sličnih veličina, CNN imaju mnogo manje veza i parametara, tako da ih je lakše trenirati, dok je njihov teoretski najbolji učinak samo nešto lošiji.

3 Konvolutivne neuronske mreže

Konvolutivne neuronske mreže predstavljaju podklasu neuronskih mreža koja ima najmanje jedan konvolutivni sloj (a može ih imati i više). Ova vrsta neuronskih mreža je inspirisana vizuelnim korteksom. Svaki put kada osoba nešto vidi, aktivira se niz slojeva neurona, i svaki sloj otkriva skup karakteristika kao što su linije, ivice itd. Viši nivoi slojeva otkrivaju složenije karakteristike kako bi prepoznali ono što je osoba videla. Konvolutivne mreže rade po istom principu i praktično su uvek *duboke neuronske mreže (eng.* deep neural network*)*[6], upravo zbog toga što je potrebno od sitnijih detalja, poput uspravnih, kosih i horizontalnih linija, koji obično bivaju detektovani u nižim slojevima mreže, konstruisati složenije oblike poput delova lica. Konvolutivne neuronske mreže se koriste u obradi signala (slike, zvuka), ali i teksta. U odnosu na ostale vrste neuronskih mreža, ističu se u prikupljanju lokalnih informacija (npr. o susednim pikselima na slici ili *,,okružujućim"(eng.* surrounding*)* rečima u tekstu) i smanjenju složenosti modela (brže treniranje, potrebno je manje uzoraka, manja šansa da dodje do *preprilagodjavanja (eng.* overfitting*)*). Konvulativne neuronske mreže se zasnivaju na sposobnosti mreža da iz sirovog signala konstruišu atribute. Nazivaju se konvolutivnim zato što uče **filtere** (pojam objašnjen u tabeli 1), čijom konvolutivnom primenom detektuju određena svojstva signala.

Postoji nekoliko različitih arhitektura u polju konvolutivnih neuronskih mreža. One se razlikuju po tipu, broju i rasporedu slojeva, i uglavnom predstavljaju bolju verziju neke od prethodnih mreža (o ovome više reči u poglavlju 3.3). Takođe, ono što treba istaći jeste podešavanje parametara mreže. Iako nije moguće izračunati u kom slučaju koji parametar treba imati koju vrednost, u narednom poglavlju (3.1) su opisani neki od bitnijih parametara koje svakako treba podesiti pri učenju mreže.

3.1 Parametri

U ovoj sekciji je dat tabelarni prikaz parametara koji su najznačajniji za implementaciju konvolutivne mreže. U tabeli 1 je dat samo kratak opis

[6]Duboke neuronske mreže su neuronske mreže koje se sastoje od više skrivenih slojeva između ulaznih i izlaznih slojeva (koji na izlazu predviđaju ciljnu promenljivu), koje mogu modelovati i kompleksne nelinearne veze.

svakog od njih radi boljeg razumevanja teksta koji sledi. Ipak, u nastavku je svaki detaljnije opisan.

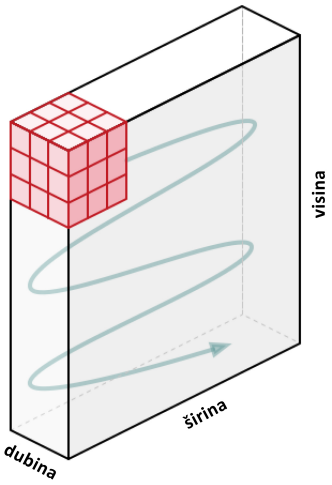
Tabela 1: Parametri konvolutivne neuronske mreže

Filter/jezgro (eng. filter/kernel*)*	- matrica sa težinama za konvoluciju ulaza - daje meru koliko deo ulaza liči na karakteristiku - tezine u matricama filtera su izvedene za vreme treniranja podataka
Proširenje (eng. Padding*)*	- koristi se za dodavanje kolona i redova nula (ili drugih vrednosti) da bi se održala konstantna veličina matrice (mape) nakon konvolucije - ovaj parametar može da unapredi performanse tako što zadržava informacije u okvirima (tim proširenjima koje dodaje)
Korak (eng. Stride*)*	- broj piksela koji se preskače dok se ulaz prelazi vodoravno i uspravno tokom konvolucije, nakon množenja svakog elementa iz ulazne matrice težina s onima u filtru
Broj kanala (eng. Number of Channels*)*	- u početku je jednak broju *kanala boja (eng.* color channels*)* ulaza, a u kasnijim stadijumima postaje jednak broju filtera koji su korišćeni za operaciju konvolucije - što je veći broj kanala, veći je i broj korišćenih filtera, a time je naučeno više karakteristika, pa postoji šansa da dođe do prilagođavanja (i obrnuto)
Parametri sloja agregacije	- ima iste parametre kao i konvolutivni sloj - uglavnom se koristi *Max-Pooling* opcija (poglavlje 3.5) - smanjuje dimenzionalnost ulazne reprezentacije (slike, izlazne matrice skrivenog sloja, itd.) tako što čuva maksimalnu vrednost podregiona

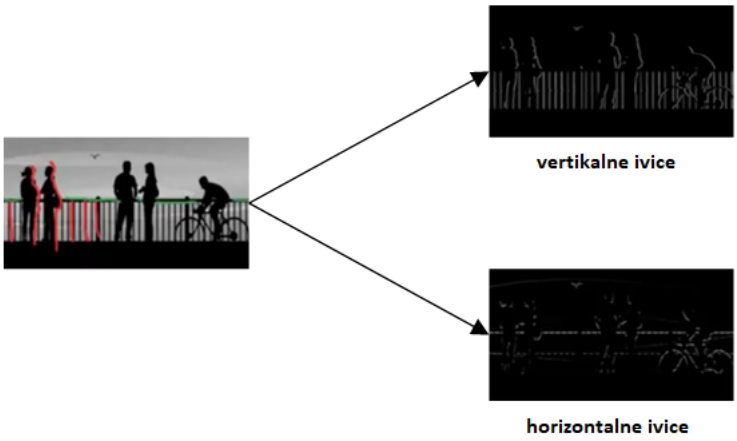
3.2 Konvolucija slika preko CNN

Da bi izvršile klasifikaciju slika, konvolutivne neuronske mreže (preciznije, konvolutivni sloj, detaljnije u poglavlju 3.4) obavljaju neku vrstu pretrage. Ovo se može zamisliti kao mali pokretni (ili klizni) prozor (prikazano na slici 2) koji klizi s leva na desno preko veće slike, i nastavlja s leve

strane kada dođe do kraja jednog prelaza (kao kod pisaće mašine). Taj pokretni prozor - koji ustvari predstavlja filter, može da prepozna samo jednu stvar, recimo kratku vertikalnu liniju (tri tamna piksela naslagana jedan na drugi). Slično, neki drugi filter može da služi za prepoznavanje horizontalnih linija, i on se takođe pomera preko piksela slike, tražeći podudaranja. Rezultat koji se postiže filterima koji prepoznaju vertikalne i horizontalne linijije prikazan je na slici 3.



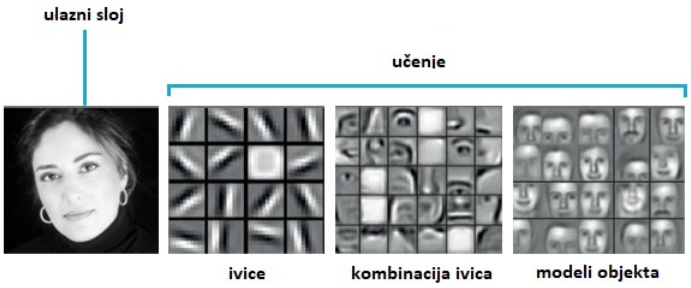
Slika 2: Kretanje filtera



Slika 3: Detektovanje vertikalnih i horizontalnih ivica

Svaki put kada dođe do poklapanja (filtera sa ulazom), ono se mapira u prostor sa karakteristikama - koji se zove ***mapa karakteristika*** *(eng.* feature maps*)*, koji je specifican za taj vizuelni element. U tom prostoru (tj. mapi) se čuva (odnosno beleži) lokacija svakog poklapanja sa vertikalnom linijom. Konvolutivna mreža pokreće mnogo pretraga nad jednom

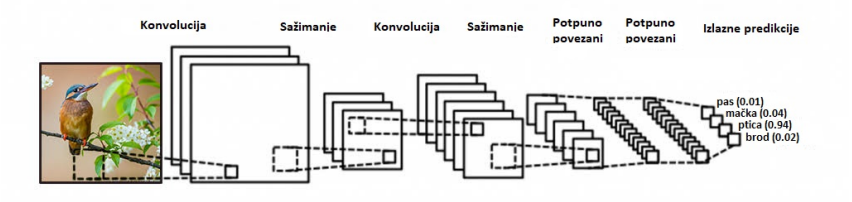
istom slikom. Na početnom sloju mreže koriste se filteri koji prepoznaju horizontalnu liniju, vertikalnu liniju i dijagonalnu liniju, da bi kreirali mapu ivica slike. U narednim koracima (odnosno slojevima) posmatraju se kombinacije ovih ivica i tako prepoznaju složeniji oblici. Ovo je demonstrirano na slici 4. U narednim poglavljima je objašnjeno iz čega se konvolutivne neuronske mreže sastoje i kako ti gradivni elementi zapravo funkcionišu.



Slika 4: Učenje mreže

3.3 Unutrasnja struktura CNN

Unutrašnja struktura konvolutivne mreže se sastoji od nekoliko naizmeničnih ***konvolutivnih slojeva*** *(eng.* convolution layer*)* i ***slojeva agregacije*** *(eng.* pooling layer*)*, pri čemu je dozvoljeno pojavljivanje iste vrste sloja više puta (prikazano na slici 5). Nakon uzastopnog smenjivanja ova dva sloja sledi ***potpuno povezani sloj*** *(eng.* fully connected layer*)*, koji se takođe može ponavljati više puta. Izlazi konvolutivnog sloja su dvodimenzionalni i nazivaju se mapama karakteristika. Oni se transformisu nelinearnom **aktivacionom funkcijom**. U konvolutivnim mrežama kao aktivacione funkcije najčešće se koriste ***prečišćena linearna jedinica*** *(eng.* Rectified Linear Unit, ReLu*)*[7] (na izlazu iz konvolutivnog sloja) i ***softmaks klasifikator*** *(eng.* softmax classifier*)* (na izlazu iz poslednjeg, potpuno povezanog, sloja mreže). U nastavku rada je svaki od tri sloja obrađen pojedinačno u poglavljima 3.4, 3.5, 3.6, dok se o aktivacionim funkcijama detaljnije govori u poglavlju 3.4.2.



Slika 5: Arhitektura konvolutivne neuronske mreže

[7]ReLu funkcija se definiše kao: f(x) = max(0,x). Prednost ReLu nad sigmnoidnom funkcijom (koja se takođe može koristiti kao aktivaciona funkcija) je što treniranje obavlja mnogo brže.

Kako su moguće različite kombinacije različitih slojeva, prirodno je da postoji više arhitektura konvolutivnih neuronskih mreža. Svaka od tih arhitektura ima svoje prednosti i mane, i u zavisnosti od problema drugačija arhitektura daje bolje (odnosno gore) rezultate. U sledećoj listi su nabrojane i ukratko opisane samo neke od najpoznatijih arhitektura konvolutivnih neuronskih mreža:

- **LeNet-5** - Ovu arhitekturu je napravio Yann LeCunn krajem 20. veka, a koristila se za čitanje zip kodova, cifara itd. LeNet-5 arhitektura se sastoji iz dva skupa konvolutivnih slojeva i slojeva agregacije, koji su praćeni *poravnavajućim konvolutivnim slojem (eng.* flattening convolutional layer*)*, za kojim slede dva potpuno-povezana sloja, a zatim softmaks klasifikator.
- **AlexNet** - Ovo je jedna od prvih dubokih neuronskih mreža. Sastoji se iz pet konvolutivnih slojeva praćena sa tri potpuno povezana sloja. Ovu mrežu je napravio Alex Krizhevsky, koji je koristio prečišćenu linearnu jedinicu - ReLu za ne-linearni deo umesto Tanh ili Sigmoid funkcije, koje su bile standardne za tradicionalne neuronske mreže.
- **VGG16** - Ova arhitektura predstavlja unapređenje arhitekture AlexNet. U njoj su veliki filteri (jezgra) zamenjeni sa više malih 3x3 filtera, jedan za drugim[8].
- **GoogLeNet** - Ova arhitektura ima čak 22 sloja. Brža je i manja od Alexnet, ali i mnogo preciznija. Ideja ove mreže jeste da se napravi model koji će moći da se koristi i na *pametnom telefonu (eng.* smartphone*)*.

3.4 Konvolutivni sloj

Konvolutivni sloj je glavni deo konvolutivne neuronske mreže koji radi najviše izračunavanja u mreži. Njegova uloga je konstrukcija novih atributa. To je prvi sloj koji ekstraktuje karakteristike iz ulazne slike. Konvolucija je matematička operacija koja uzima dva ulaza - dve matrice f i g, dimenzija m x n i p x q, a definisana je na sledeći način:

$$(f * g)_{ij} = \sum_{k=0}^{p-1} \sum_{l=0}^{q-1} f_{i-k,i-l} g_{k,l}$$

Matrica f predstavlja ulaz, poput slike, a matrica g filter. Određena ulazna reprezentacija podatka (originalna slika) konvolvira se sa filterom sa određenim parametrima (ti parametri predstavljaju i parametre konvolutivne neuronske mreže). Procesom učenja ovi parametri (težine koje je potrebno naučiti kako bi mreža davala dobre rezultate) se podešavaju i bivaju naučeni. Filteri su najcešce manjih prostornih dimenzija od ulaza, ali uvek su jednake dubine kao i ulaz. Kao što je vec rečeno, konvolucijska jezgra (filteri) filtriraju sliku, tj. mapu karakteristika kako bi dobili neku korisnu informaciju kao što je recimo određeni oblik, boja ili ivica.

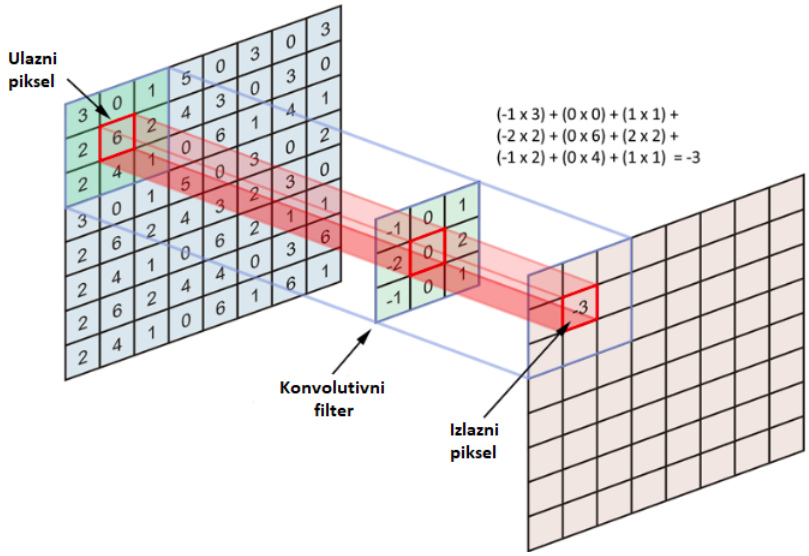
Tokom prvog prolaza svaki filter se pomera po širini i visini ulaza i računa se skalarni proizvod ulaza i vrednosti filtera (prikazano u formuli iznad). Ako obe matrice imaju visoke vrednosti na istim pozicijama, onda će i vrednost skalarnog proizvoda biti velika. A, ako nemaju, onda će i

[8]Više manjih filtera se pokazuje boljim nego jedan veliki filter (jezgro), zato što više nelinearnih slojeva povećavaju dubinu mreže što im omogućava da uče kompleksnije karakteristike (i to po manjoj ceni).

vrednost skalarnog proizvoda biti mala. Na taj način, samo na osnovu te vrednosti, može se zaključiti da li sadržaj slike odgovara sadržaju koji filter traži.

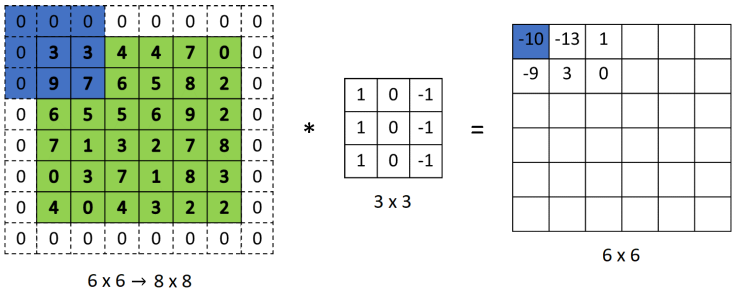
Izlaz jednog filtera je dvodimenzioni niz. Ako postoji više filtera, izlaz iz konvolutivnog sloja su rezultati svih filtera (mape karakteristika) poredjani po dubini. Broj mapa karakteristika na izlazu, odnosno broj primenjenih filtera, predstavlja **dubinu mreže**. Operacijom konvolucije dobija se transformirana slika dimenzije I*K, gde je I dimenzija ulazne matrice slike, a K dimenzija filtera koji je primenjen nad tom slikom. Takođe, dobijena transformacija je lokalna tj. pikseli izlaza zavise od lokalnih, susednih piksela ulaza. Ceo proces konvolucije, za koji je upotrebljen filter dimenzije 3x3 i ulazna matrica dimenzije 8x8, prikazan je na slici 6. U podpoglavljima 3.4.1 i 3.4.2 su opisani parametri koji bitno utiču na ishod rada mreže.

Slika 6: Konvolucija primenom filtera dimenzije 3x3 na matricu dimenzije 8x8



3.4.1 Proširivanje

Formula konvolucije koja je data u poglavlju 3.4 nije definisana za sve indekse i = $\overline{0, m-1}$ i j = $\overline{0, n-1}$. Na primer, ako je i, j = 0 i k, l > 0, vrednost $f_{i-k,i-l}$ nije definisana. Ukoliko bi se u obzir uzele samo definisane vrednosti, dimenzija konvolucije bi bila manja od dimenzije matrice f. Međutim, to nije uvek poželjno, i može se izbeći tako što se vrši **proširivanje** matrice f, na primer nulama ili vrednostima koje su već na obodu, tako da veličina rezultujuće matrice bude jednaka veličini matrice f pre proširivanja. Ovo je prikazano na slici 7. Takođe, prilikom računanja konvolucije, filter se duž slike ne mora pomerati za jedan piksel, već za neki veći **korak** (pojam opisan u tabeli 1).



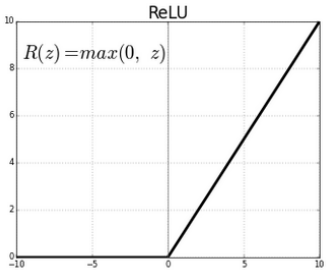
Slika 7: Primer proširivanja ulazne matrice nulama

3.4.2 Aktivaciona funkcija

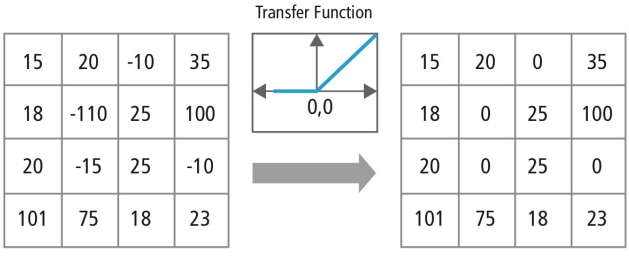
U konvolutivnim slojevima kao aktivaciona funkcija se najčešće koristi ReLu (Rectified Linear Unit) funkcija. Definisana je kao

$$f(z) = max(0, z).$$

To znači da će negativne piksele mape karakteristika nastale konvolucijom da postavi na nulu i neće aktivirati odgovarajuće neurone. A pozitivne piksele neće menjati. Na taj način, ReLu ne aktivira sve neurone u istom trenutku, čime ubrzava rad mreže i čini je efikasnijom. Na slici 8 se može videti kako ReLU funkcija izgleda, a na slici 9 kako se pomoću nje transformiše mapa karakteristika.



Slika 8: Izgled ReLU funkcije



Slika 9: Primer rada ReLU funkcije nad mapom karakteristika

3.5 Sloj agregacije

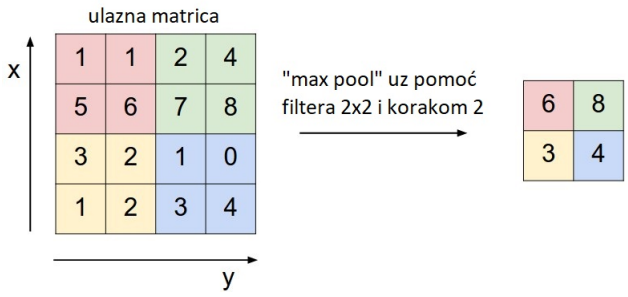
Uloga sloja za agregaciju je smanjenje broja parametara kada su slike prevelike, kao i smanjenje broja računskih operacija u višim slojevima mreže. Agregacija smanjuje dimenzionalnost svake mape, ali zadržava važne informacije. Sve to rezultuje smanjenjem računske zahtevnosti pri optimizaciji i pomaže u kontroli *overfitinga*. Zato je poželjno da slojevi za agregaciju prate konvolutivne slojeve kako bi postepeno smanjili prostornu veličinu (širinu i visinu) prikaza podataka.

Često konačni zadatak postavlja neko globalno pitanje o slici, na primer, da li sadrži mačku? Stoga, čvorovi zadnjeg sloja moraju biti osetljivi na celi ulaz. Postepenom agregacijom informacija, stvarajući sve grublje i grublje mape karakteristika, postiže se taj cilj da se na kraju nauči globalna reprezentacija, zadržavajući sve prednosti konvolutivnih slojeva na srednjim slojevima obrade.

Sloj agregacije ukrupnjuje informacije, tako sto računa neku jednostavnu funkciju agregacije susednih jedinica prethodnog sloja, poput *maksimuma (eng.* Max pooling*)*, koji vraća maksimalnu vrednost dela slike pokrivene filterom, ili *proseka (eng.* Average pooling*)*, koji vraća prosečnu vrednost dela slike pokrivene filterom. Ukoliko agregira, na primer, 3 x 3 piskela, onda je broj izlaza ovog sloja 9 puta manji od broja izlaza prethodnog. Kada se računa maksimum, dolazi do zanemarivanja informacija o tome gde je precizno neko svojstvo (poput uspravne linije) pronađeno, ali se ne gubi informacija da je ono pronađeno. Ovakva vrsta zanemarivanja informacije često ne šteti cilju koji treba postići. Na primer, ako su na slici pronađeni kljun i krila, informacija o tačnoj poziciji najverovatnije nije bitna za odlučivanje da li se na slici nalazi ptica. Ipak, ukoliko je potrebno napraviti mrežu koja igra igru u kojoj su pozicije objekata na ekranu bitne, nije poželjno koristiti agregaciju.

Prva slika (pool): In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved.

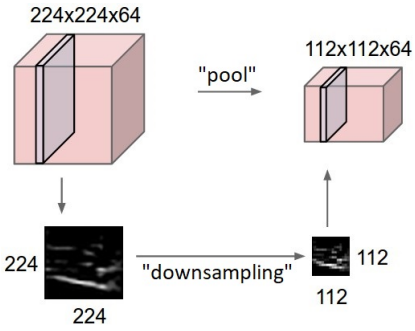
Druga slika (maxpool): The most common downsampling operation is max, giving rise to max pooling, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).



Slika 10: Maxpool

3.6 Potpuno povezani sloj

Nakon što su u konvolutivnom i agregativnom sloju sakupljene informacije o slici, sledeće što je potrebno uraditi jeste zaključiti kojoj klasi



Slika 11: Pool

pripada ta slika. Za to se koristi jedan ili više potpuno povezanih slojeva (eng. *fully connected layer* ili *dense layer*).

U potpuno povezanom sloju, svaki neuron je povezan sa svakim neuronom iz prethodnog sloja, kao što je to slučaj kod običnih neuronskih mreža. Taj sloj kao ulaz očekuje vektor, što znači da je prvo potrebno ispraviti (eng. *flatten*) mapu karakteristika, dobijenu nakon konvolucije i agregacije, u vektor. U svim slojevima, osim u poslednjem, može da se koristi ReLU aktivaciona funkcija.

Poslednji potpuno povezani sloj treba da ima onoliko neurona koliko ima i klasa, kako bi mogao da odredi koju klasu da pridruži slici. U ovom sloju je najbolje koristiti softmax aktivacionu funkciju. Softmax funkcija određuje verovatnoću pripadnosti slike svakoj od klasa. Nakon njene primene, dobija se vektor vrednosti koje odgovaraju verovatnoćama i koje u zbiru daju vrednost 1. Zahvaljujući njoj, slika se klasifikuje u onu klasu kojoj odgovara najveća verovatnoća pripadnosti. Softmax funkcija je definisana kao:

$$S(z_i) = \frac{e_i^z}{\sum_{j=1}^{K} e_j^z}, i = 1, ..., K, z = (z_1, ..., z_K)$$

gde je z ulazni vektor od K realnih brojeva.

U sloju koji koristi softmax aktivacionu funkciju, obično se za funkciju određivanja greške klasifikacije uzima *categorical crossentropy*. Ona se koristi ukoliko su podaci takvi da jedna instanca tih podataka može da pripada samo jednoj klasi. *Categorical crossentropy* upoređuje raspodelu verovatnoća predviđenih klasa sa raspodelom koja odgovara pravim klasama. Ova funkcija je definisana kao:

$$H(y, \hat{y}) = -\sum_i y_i \cdot log(\hat{y}))$$

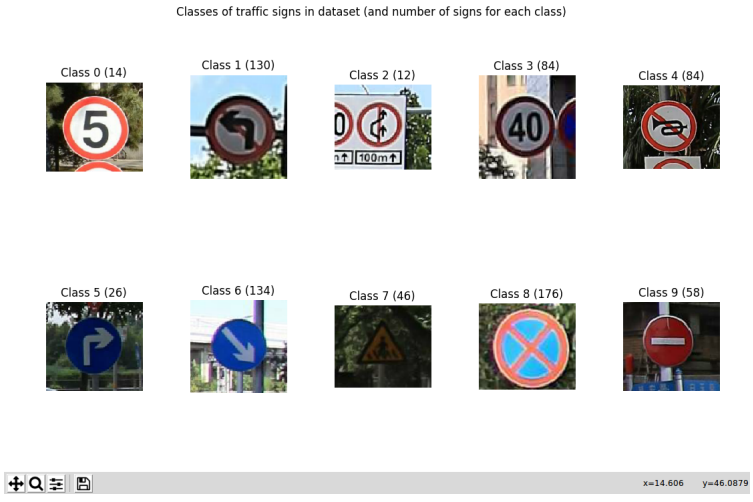
gde y predstavlja vektor pravih raspodela, a $\hat{y}$ vektor predviđenih raspodela.

4 Implementacija i eksperimentalni rezultati

Neuronsku mrežu smo implementirale u programskom jeziku Python korišćenjem Keras biblioteke. Kao podatke za trening i testiranje, koristile

smo slike iz baze podataka kineskih saobraćajnih znakova [1]. Baza sadrži 6164 slika saobraćajnih znakova podeljenih u 58 klasa, pri čemu trening skup sadrži 4170, a test skup 1994 slike. Međutim, nisu sve klase imale podjednak broj slika. Za neke je postojalo 5 slika na kojima bi mreza mogla da trenira, a za neke oko 400. Takođe, zbog prevelike količine podataka, izvršavanje programa je bilo mnogo sporo. Zbog svega toga, odlučile smo da koristimo samo jedan deo te baze i izdvojile 10 klasa koje su imale približno jednak broj slika. Nakon toga, dobile smo trening skup od 1693 slika i test skup od 764 slike. Na slici 12 je prikazan po jedan znak iz svake klase trening skupa, zajedno sa brojem elemenata te klase. Slično, ti podaci o test skupu, mogu se videti na slici 13.

Slika 12: Trening skup



Slika 13: Test skup

U narednim poglavljima će biti predstavljeni neki od modela konvolutivnih neuronskih mreža koje smo pravile, a koje su davale najbolje rezultate.

4.1 Model 1

Jedan od prvih modela koji je imao veći uspeh na test podacima prikazan je u kodu 1. Sastoji se iz četiri konvolutivna, dva agregativna i dva potpuno povezana sloja. Detaljna arhitektura mreže se može videti na slici 14. U svim konvolutivnim slojevima veličina jezgra je 3x3, a broj filtera na izlazu iz konvolucije je 32. Takođe, u svakom se koristi ReLU aktivaciona funkcija. U sloju agregacije, sažimanje se vrši biranjem maksimalne vrednosti dela mape karakteristika koji je prekriven filterom. Kako ne bi došlo do preprilagodjavanja modela trening podacima, povremeno je, pomoću funkcije Dropout(), isključivan određen broj nasumično odabranih neurona. Nakon agregacija je isključeno 20% neurona, a pre poslednjeg, potpuno povezanog, sloja 50%. Poslednji potpuno povezani sloj ima onoliko neurona koliko ima klasa i koristi softmax aktivacionu funkciju.

Učenje modela je sprovedeno u 30 epoha. Batch size je postavljen na 32, što znači da će u svakoj iteraciji biti uzeta 32 primerka iz trening skupa koja će biti propagirana kroz mrežu. Optimizacija modela je izvršena pomoću gradijentnog spusta. Takođe, pre početka treninga, sve slike su skalirane na istu veličinu, 64x64 piksela.

```
def cnn_model():

    model = Sequential()

    model.add(Conv2D(filters = 32, kernel_size = (3, 3),
        padding='same', input_shape=(IMG_SIZE, IMG_SIZE, 3),
        data_format="channels_last", activation='relu'))
    model.add(Conv2D(filters = 32, kernel_size = (3, 3),
        activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(filters = 32, kernel_size = (3, 3),
        padding='same', activation='relu'))
    model.add(Conv2D(filters = 32, kernel_size = (3, 3),
        activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(NUM_OF_CLASSES, activation='softmax'))

    model.summary()
    return model


batch_size = 32
epochs = 30
lr = 0.01   #learning rate

model = cnn_model()

# optimizacija pomocu gradijentnog spusta
sgd = SGD(lr=lr, decay=1e-6, momentum=0.9, nesterov=True)

model.compile(loss='categorical_crossentropy', optimizer=sgd,
        metrics=['accuracy'])

def lr_schedule(epoch):
```

```
    return lr * (0.1 ** int(epoch / 10))

model.fit(images, classes,
          batch_size=batch_size,
          epochs=epochs,
          validation_split=0.2,
          callbacks=[LearningRateScheduler(lr_schedule),
                     ModelCheckpoint('model.h5', save_best_only=True)])
```

Kod 1: Model 1

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 64, 64, 32)        896
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 62, 62, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 31, 31, 32)        9248
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 29, 29, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 14, 14, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               3211776
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
=================================================================
Total params: 3,245,546
Trainable params: 3,245,546
Non-trainable params: 0
```

Slika 14: Slojevi modela neuronske mreže

Na slici 15 možemo videti da je preciznost modela na trening skupu tokom poslednje tri epohe varirala oko 0.99 i 1.0. Na slici 16 možemo videti kako se model pokazao na test skupu. Preciznost (*accuracy*, tj. broj tačno klasifikovanih instanci / ukupan broj instanci) je jednaka 0.882. Vidimo da je i preciznost (*precision*) koja se odnosi na svaku od klasa dosta visoka (osim za klase 0 i 5)

```
Epoch 28/30
1311/1311 [==============================] - 110s 84ms/step - loss: 0.0046 - acc: 0.9985 - val_loss: 4.6311e-04 - val_acc: 1.0000
Epoch 29/30
1311/1311 [==============================] - 103s 78ms/step - loss: 0.0014 - acc: 1.0000 - val_loss: 4.4722e-04 - val_acc: 1.0000
Epoch 30/30
1311/1311 [==============================] - 110s 84ms/step - loss: 0.0035 - acc: 0.9992 - val_loss: 4.3213e-04 - val_acc: 1.0000
```

Slika 15: Preciznost na trening podecima

```
Test accuracy = 0.8821989528795812

Classification report:
              precision    recall  f1-score   support

           0       0.50      0.86      0.63        14
           1       0.96      0.78      0.86       130
           2       1.00      0.83      0.91        12
           3       0.91      0.93      0.92        84
           4       0.91      0.95      0.93        84
           5       0.67      0.46      0.55        26
           6       1.00      0.81      0.89       134
           7       0.76      0.96      0.85        46
           8       0.84      0.99      0.91       176
           9       0.93      0.93      0.93        58

   micro avg       0.88      0.88      0.88       764
   macro avg       0.85      0.85      0.84       764
weighted avg       0.90      0.88      0.88       764

Confusion matrix:
[[ 12   0   0   0   0   0   2   0   0]
 [  6 102   0   0   8   4   0   6   0   4]
 [  0   0  10   2   0   0   0   0   0   0]
 [  2   2   0  78   0   0   0   2   0   0]
 [  2   0   0   0  80   0   0   2   0   0]
 [  0   0   0   0   0  12   0   0  14   0]
 [  2   0   0   4   0   0 108   0  20   0]
 [  0   0   0   2   0   0   0  44   0   0]
 [  0   0   0   0   0   2   0   0 174   0]
 [  0   2   0   0   0   0   0   2   0  54]]
```

Slika 16: Preciznost na test podecima

4.2 Model 2: LeNet-5

Još bolje rezultate smo dobile pomoću LeNet-5 mreže [2]. LeNet-5 mrežu je 1990ih napravio Yann LeCunn. Mreža se sastoji iz 7 slojeva, pri čemu se ne računa ulazni sloj. Sve slike smo skalirale na večičinu 32x32 piksela, jer ova mreža kao ulaz očekuje slike tih dimenzija. Prvi konvolutivni sloj na izlazu daje 6 mapa karakteristika, koristi jezgro veličine 3x3 i kao aktivacionu funkciju koristi ReLU. Nakon njega sledi sloj agregacije u kom se sažimanje vrši biranjem prosečne vrednosti dela mape karakteristika koji je prekriven filterom. Još jednom se ponavljaju ova dva sloja, s tim što sada konvolutivni sloj vraća 16 mapa karakteristika. Nakon ispravljanja mape karakteristika u vektor, taj vektor se prosleđuje potpuno povezanom sloju koji se sastoji iz 120 neurona i koristi ReLU aktivacionu funkciju. Nakon toga sledi potpuno povezani sloj od 84 neurona koji, takođe, koristi ReLU aktivacionu funkciju. I, na kraju, ostaje još jedan potpuno povezani sloj koji ima onoliko neurona koliko ima klasa i koji koristi softmax aktivacionu funkciju. U kodu 2 je data implementacija LeNet-5 mreže, a na slici 17 je prikazana njena detaljna arhitektura.

```
def cnn_model():

    model = Sequential()

    model.add(Conv2D(filters=6, kernel_size=(3, 3),
        activation='relu', input_shape=(IMG_SIZE, IMG_SIZE, 3),
        data_format="channels_last"))
    model.add(AveragePooling2D())

    model.add(Conv2D(filters=16, kernel_size=(3, 3),
        activation='relu'))
    model.add(AveragePooling2D())
```

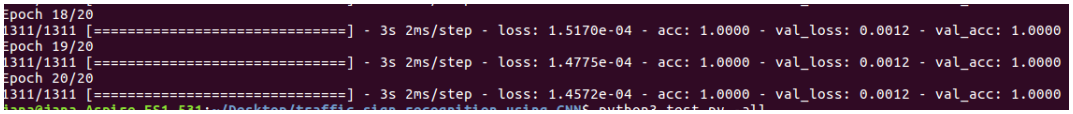
```
    model.add(Flatten())
    model.add(Dense(units=120, activation='relu'))
    model.add(Dense(units=84, activation='relu'))
    model.add(Dense(units=NUM_OF_CLASSES, activation = 'softmax'))

    model.summary()
    return model


batch_size = 32
epochs = 20
lr = 0.01

model = cnn_model()

model.compile(loss=keras.losses.categorical_crossentropy,
        optimizer=keras.optimizers.Adam(),
        metrics=['accuracy'])

def lr_schedule(epoch):
    return lr * (0.1 ** int(epoch / 10))

model.fit(images, classes,
          batch_size=batch_size,
          epochs=epochs,
          validation_split=0.2,
          callbacks=[LearningRateScheduler(lr_schedule),
                 ModelCheckpoint('model.h5', save_best_only=True)])
```

Kod 2: LeNet-5

```
Using TensorFlow backend.
Number of classes: 10
Number of images for training: 1639
classes shape:  (1639, 10)
images shape:  (1639, 32, 32, 3)
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 30, 30, 6)         168
_________________________________________________________________
average_pooling2d_1 (Average (None, 15, 15, 6)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 13, 16)        880
_________________________________________________________________
average_pooling2d_2 (Average (None, 6, 6, 16)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_1 (Dense)              (None, 120)               69240
_________________________________________________________________
dense_2 (Dense)              (None, 84)                10164
_________________________________________________________________
dense_3 (Dense)              (None, 10)                850
=================================================================
Total params: 81,302
Trainable params: 81,302
Non-trainable params: 0
_________________________________________________________________
Train on 1311 samples, validate on 328 samples
```

Slika 17: Slojevi modela leNet-5 neuronske mreže

Na slici 18 možemo videti da je preciznost na trening skupu u poslednje tri iteracije bila najveća moguća, tj 1.00. Na slici 19 je prikazan rezultat rada modela na test podacima. Preciznost (*accuracy*) iznosi 0.91. Takođe vidimo da je preciznost (*precision*) za pojedinačne klase visoka. Na osnovu matrice konfuzije može se videti da veoma malo podataka pogrešno klasifikuje.



Slika 18: Preciznost na trening podecima

```
Test accuracy = 0.9109947643979057

Classification report:
              precision    recall  f1-score   support

           0       0.86      0.86      0.86        14
           1       0.82      1.00      0.90       130
           2       1.00      1.00      1.00        12
           3       0.87      0.93      0.90        84
           4       1.00      0.55      0.71        84
           5       0.71      0.92      0.80        26
           6       0.98      0.93      0.95       134
           7       0.85      1.00      0.92        46
           8       0.97      0.97      0.97       176
           9       1.00      0.93      0.96        58

   micro avg       0.91      0.91      0.91       764
   macro avg       0.91      0.91      0.90       764
weighted avg       0.92      0.91      0.91       764

Confusion matrix:
[[ 12   0   0   0   0   0   0   2   0   0]
 [  0 130   0   0   0   0   0   0   0   0]
 [  0   0  12   0   0   0   0   0   0   0]
 [  0   4   0  78   0   2   0   0   0   0]
 [  0  24   0   8  46   0   0   2   4   0]
 [  0   0   0   0   0  24   0   0   2   0]
 [  0   0   0   4   0   6 124   0   0   0]
 [  0   0   0   0   0   0   0  46   0   0]
 [  2   0   0   0   0   2   2   0 170   0]
 [  0   0   0   0   0   0   0   4   0  54]]
```

Slika 19: Preciznost na test podecima

5 Zaključak

Literatura

[1] a. Chinese Traffic Sign Database. on-line at: http://www.nlpr.ia.ac.cn/pal/trafficdata/recognition.html.

[2] Leon Bottou Yann LeCun, Patric Haffner. Object recognition with Gradient-Based learning. 1998.

A Dodatak

A.1 Podnaslov 1

Dodatna objasnjenja

By visualizing the output from different convolution layers in this manner, the most crucial thing that you will notice is that the layers that are deeper in the network visualize more training data specific features, while the earlier layers tend to visualize general patterns like edges, texture, background etc.

This knowledge is very important when you use Transfer Learning whereby you train some part of a pre-trained network (pre-trained on a different dataset, like ImageNet in this case) on a completely different dataset. The general idea is to freeze the weights of earlier layers, because they will anyways learn the general features, and to only train the weights of deeper layers because these are the layers which are actually recognizing your objects.

Convolved Feature, Activation Map or Feature Map is the output volume formed by sliding the filter over the image and computing the dot product. Perceptrons come first in 1950s, and it uses a brittle activation function to do classification, so if w*x is greater than some value it predicts positive, otherwise negative.

Neurons uses a softer activation function by introducing a sigmoid function, a tanh function or other activation functions to pass on values to other neurons in the network.

So perceptrons do not use in a network setting, they do classification on their own, hence they can't classify XOR, however neurons can because they all contribute forward to the final output, using more complicated structure(i.e. multiple layers in network), they are able to classify XOR and other complicated problems.

A CNN, in specific, has one or more layers of convolution units. A convolution unit receives its input from multiple units from the previous layer which together create a proximity. Therefore, the input units (that form a small neighborhood) share their weights.

The convolution units (as well as pooling units) are especially beneficial as:

They reduce the number of units in the network (since they are many-to-one mappings). This means, there are fewer parameters to learn which reduces the chance of overfitting as the model would be less complex than a fully connected network. They consider the context/shared information in the small neighborhoods. This future is very important in many applications such as image, video, text, and speech processing/mining as the neighboring inputs (eg pixels, frames, words, etc) usually carry related information.