Univerzitet u Zenici
Politehnički fakultet
Odsjek: Softversko Inženjerstvo

SAVREMENI OBRASCI KONVOLUCIONIH NEURONSKIH MREŽA

- Seminarski rad iz predmeta Rudarenje Podataka -

Kandidati:
Selimanović Harun
Smajlović Admir

Mentori:
Doc.dr. Dželihodžić Adnan
Ass. Mujić Ahmed
S.i.p Hambo Faris

Zenica 2025

SADRŽAJ

1. UVOD

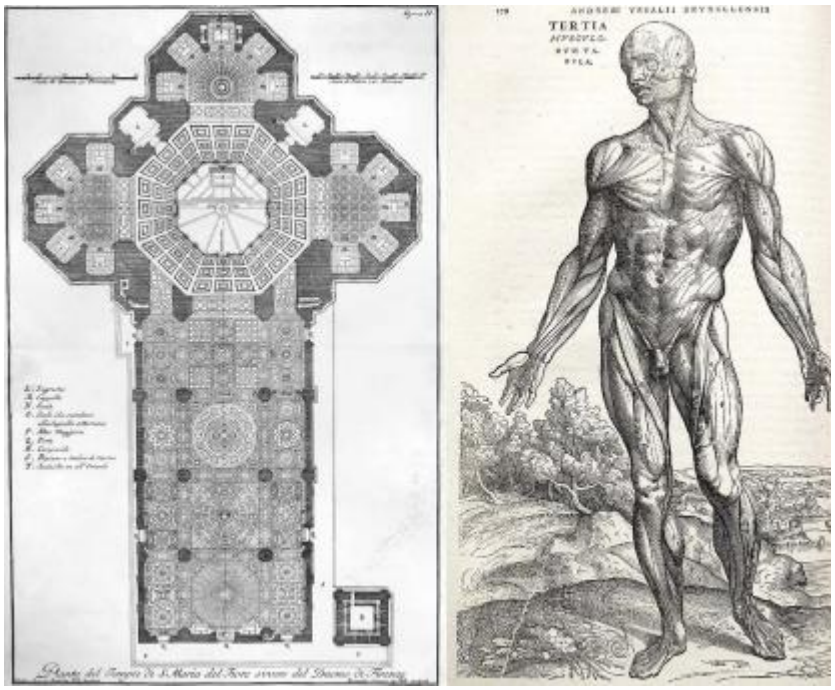
„Arhitektura“ modela predstavlja zbir svih odluka koje su donesene prilikom njegovog stvaranja: koje slojeve da koristimo, kako ih konfigurisati i na koji način ih međusobno povezati. Ove odluke definišu hipotezijski prostor modela. Hipotezijski prostor je prostor mogućih funkcija koje gradijentni spust može pretraživati i parametarski je definisan težinama modela. Dobar hipotezijski prostor uključuje prethodno znanje koje se ima o konkretnom problemu i njegovom rješenju. Da bi se učinkovito učilo iz podataka potrebno je napraviti određene pretpostavke o tome šta se traži.

„Arhitektura“ modela često čini razliku između uspjeha i neuspjeha. Ako se naprave neprikladni arhitektonski izbori, model može da ima loše rezultate i nijedna količina podataka za treniranje ga ne može spasiti. Nasuprot tome, dobra arhitektura modela je ona koja ubrzava učenje i omogućava modelu da efikasno koristi dostupne podatke za treniranje i time smanjuje potrebu za velikim skupovima podataka.

„Arhitektura“ modela je više umjetnost nego nauka. Iskusni inžinjeri mašinskog učenja su sposobni da „sastave“ modele visoke tačnosti već iz provog pokušaja, dok se početnici muče da naprave model koji se uopšte može trenirati. Ključna riječ je intuicija. Niko ne može dati tačan odgovor šta funkcioniše a šta ne funkcioniše. Stručnjaci se oslanjaju na prepoznavanje obrazaca, sposobnost koju stiču kroz veliko praktično iskustvo. Međutim nije ni sve u intuiciji, iako nema puno nauke u ovome području kao i u svakom inženjerskom poslu postoje određene najboje prakse.

2. MODULARNOST, HIJERARHIJA I PONOVNO KORIŠTENJE

Ako se želi pojednostaviti složen sistem, postoji univerzalni način koji se može primjeniti a to je da se jednostavno sva ta kompleksnost podijeni na module. Zatim da se ti moduli organizuju hijerarhijski i na kraju iskoristiti te module na više mjesta gdje god da je to potrebno. To se naziva MHR formulom (eng. Modularity – Hierarchy – Reuse) i to je osnovna arhitektura sistema u gotovo svakoj oblasti u kojoj se koristi pojam „arhitektura“. To je sama srž organizacije svakog sistema značajne složenosti bilo da se to radi o katedrali, ljudskom tijelu ili Keras biblioteci. Primjer tih modula je prikazan na sljedećoj slici:

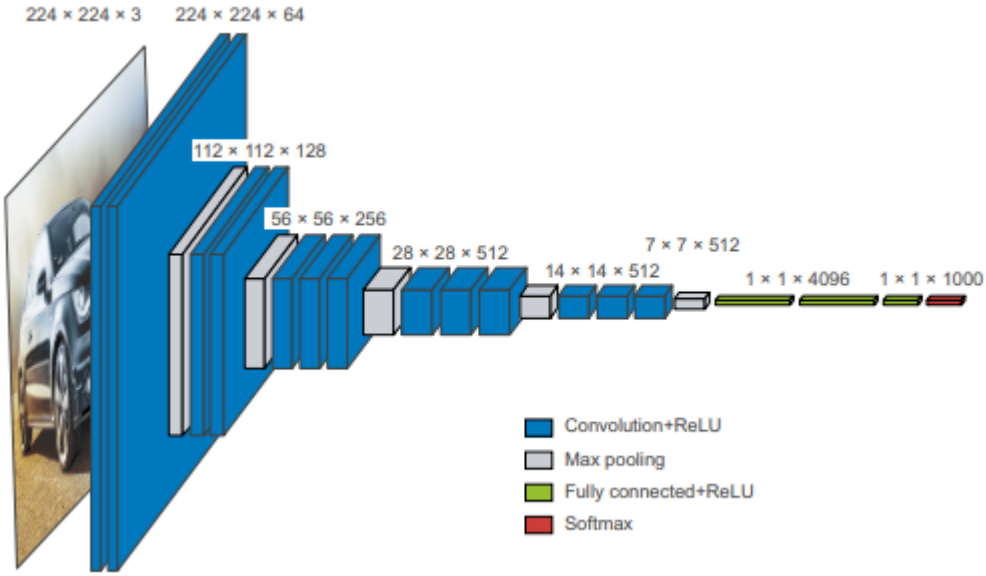


Slika 1: Primjer kompleksnih sistema koji su podijeljeni na module (Katedrala je podijeljena na više prostorija i kupola koji čine module, dok kod ljudskog tijela svaki prst predstavlja modul)

Softverski inžinjeri trebaju biti upoznati sa ovim principima: efikasan kod je onaj kod koji je modularan, hijerarhijski i u kojem se ne implementira ista stvar više puta već se umjesto toga koriste klase i funkcije za osnovnu upotrebu. Ako se kod organizuje prateći ove principe može se reći da se primjenjuje softverska arhitektura.

Pojam dubokog učenja je u suštini primjena ovog principa te kontinuiranu optimizaciju pomoću gradijent spusta. Uzmimo klasičnu tehniku optimizacije (gradijent spust u prostoru kontinuiranih funkcija) i struktuirajmo prostor pretrage u module (slojeve) koji su organizovani u duboku hijerarhiju pri čemu se ponovo koristi ono što može. Npr. konvolucije su zasnovane na ponovnoj upotrebi iste informacije na različitim prostornim lokacijama.

Arhitektura modela dubokog učenja prvenstveno se zasniva na pametnoj primjeni modularnosti, hijerarhiji i ponovnoj upotrebi (MHR). Sve popularne konvolucijske neuronske mreže nisu samo struktuirane po slojevima nego i u ponavljajućim grupama slojeva (tzv. blokovi ili moduli). Većina konvolucijskih mreža ima piramidalnu strukturu (tzv. hijerarhiju osobina / karakteristika)



Slika 2: Primjer ponavljajućih blokova i piramidalne strukture mape karakteristika

Na slici vidimo primjenu modularnosti i korištenja blokova te također vidimo kako se implementira i hijerarhija. Rani slojevi detektuju osnovne oblike (ivice, linije) a kasniji slojevi detektuju komleksinije strukture kao što je u ovome slučaju automobil. Također vidimo kako se isti tipovi blokova ponavljaju što pojednostavljuje dizajn.

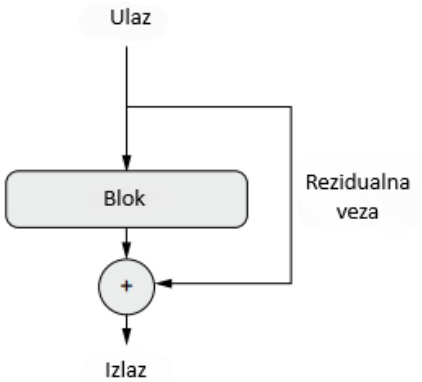
Dublje hijerarhije su same po sebi korisne jer podstiču ponovnu upotrebu osobina a time i apstrakciju. Općenito dubok niz uskih slojeva daje bolje rezultate nego plitak niz širokih slojeva. Međutim postoji granica koliko duboko se mogu slagati slojevi, zbog problema nestajućih

gradijenata. To nas dovodi do prvog ključnog obrasca u arhitekturi modela a to su rezidualne veze.

3. REZIDUALNE VEZE

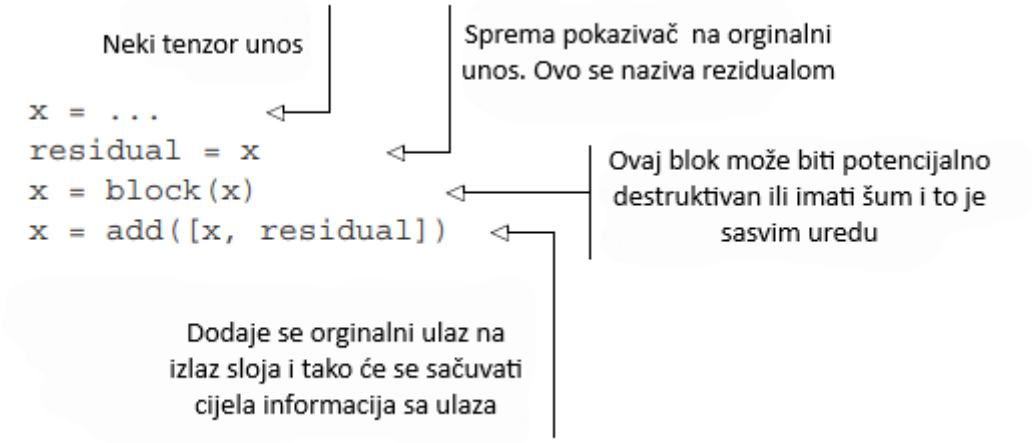
Poznata igra gluhih telefona, gdje se početna poruka šapne u uho jednom igraču koji je zatim šapne narednom igraču i tako redom. Na kraju, završna poruka veoma malo liči na orginalnu verziju. Ovo je metafora za kumulativne greške koje nastaju pri sekvencijalnom prijenosu kroz kanale. Ispostavlja se da je backpropagation (propagacija greške unazad) u sekvencijalnom modelu dubokog učenja prilično sličan igri gluhih telefona.
Postoji lanac funkcija y = f4 (f3 (f2 (f1 (x)))) i cilj je da se prilagode svi parametri funkcije u tom lancu na osnovu greške zabilježene na izlazu funkcije f4. Da bi se prilagodio f1 moraju se informacije o greščci prenijeti unazad kroz f4, f3, f2. Međutim svaka naredna funkcija u lancu dovodi do određenog nivoa šuma. Ako je lanac previše dubok taj šum počinje da preplavljuje korisne informacije o gradijentu i backpropagation prestaje da funkcioniše. Ovaj tip modela uopšte ne može da uči i ovo je poznato kao problem nestajućih gradijenata (eng. Vanishing gradients). Rješenje ovog problema je jednostavno. Potrebno je „natjerati“ svaku funkciju u lancu da sačuva čistu verziju informacije sadržane u prethodnom ulazu. Najlakši način da se to postigne je korištenje rezidualne veze (eng. residual connection) tako da se ulazni sloj ili blok slojeva doda na njegov izlaz.

Rezidualna veza djeluje kao prečica za informaciju zaobilazeći destruktivne ili šumne blokove. Na taj način informacija o grešci iz izlaza modela može prenosti bez šuma nazad kroz duboku mrežu sve do početnih slojeva. Ova tehnika je predstavljena 2015-te godine.



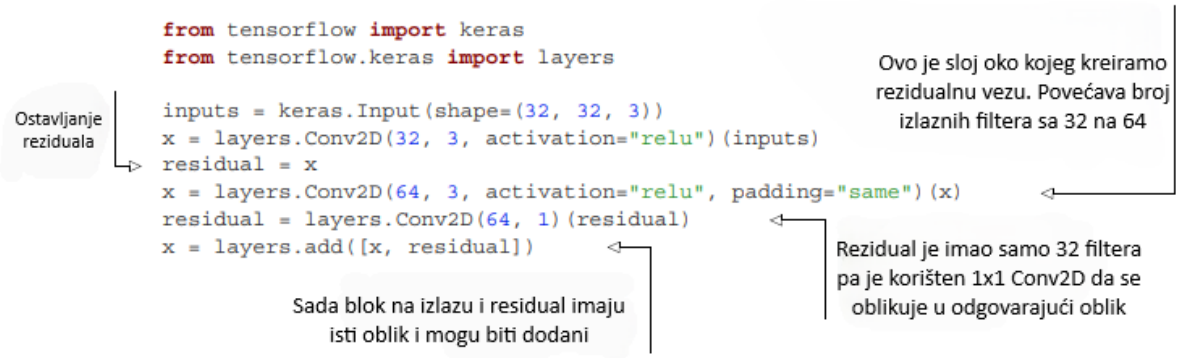
Slika 3:Primjer rezidualne veze

Praktični primjer:

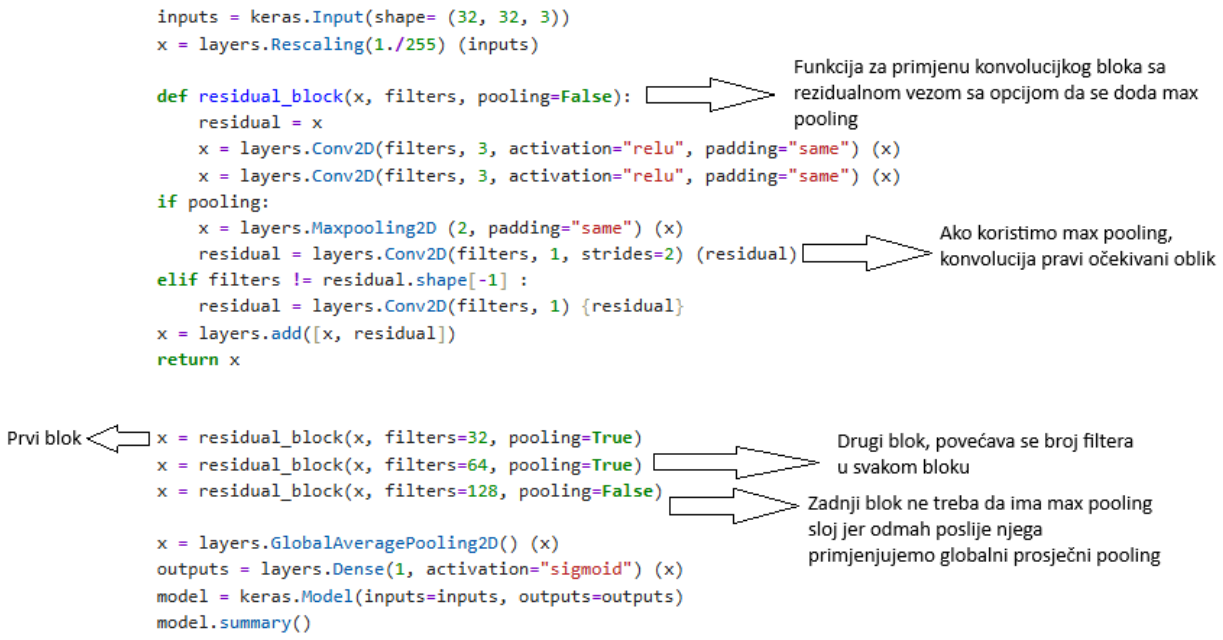


Treba imati na umu da dodavanje ulaza na izlaz bloka podrazumijeva da izlaz treba imati isti oblik kao i ulaz. Međutim to nije slučaj ako blok sadrži konvolucijske slojeve sa povećanim brojem filtera ili max pooling sloj. U takvim slučajevima se koristi Conv2D sloj dimenzija 1x1 bez aktivacijske funkcije i time se rezidualna putanja linearno projektuje na željeni oblik izlaza.

Primjer rezidualnog bloka gdje se broj filtera mijenja:



Da bi ove ideje učinili konkretnijim u nastavku je pokazan primjer jednostavne konvolucijske mreže koja je struktuirana iz niza blokova pri čemu je svaki blok sastavljen od dva konvolucijska sloja i jednog opcionalnog sloja za max pooling sas rezidualnom vezom oko svakog bloka.



Uz rezidualne veze moguće je graditi mreže proizvoljne dubine bez potrebe da se brine o nestajućim gradijentima.

4. BATCH NORMALIZACIJA

Normalizacija je široka kategorija modela koji imaju za cilj da omoguće različitim uzorcima da su vidljivi od strane modela mašinskog učenja te da su međusobno slični. To pomaže modelu da bolje uči i da generalizuje nove podatke. Najčešći oblik normalizacije podataka je onaj koji centrira podatke oko nule oduzimanjem srednje vrijednosti od podataka i davanje podacima standardnu devijaciju dijeljenjem sa standardnom devijacijom. U suštini ovo podrazumijeva pretpostavku da podaci prate normalnu (ili Gausovu) raspodjelu i osigurava da je ta raspodjela centrirana i skalirana na jediničnu varijansu:

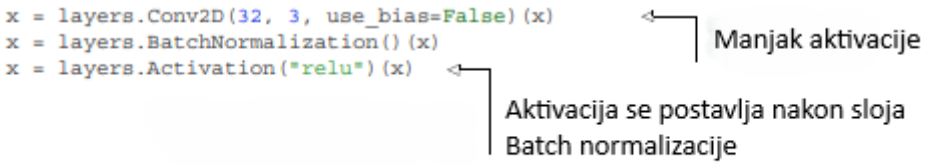
normalized_data = (data - np.mean(data, axis=...)) / np.std(data, axis=...)

Normalizacija podataka može biti korisna i nakon svake transformacije koju mreža izvrši čak i ako podaci koji ulaze u Dense ili Conv2D mrežu imaju srednju vrijednost 0 i standardnu devijaciju 1. Ne postoji razlog da se unaprijed pretpostavi da će to biti i slučaj za podatke koji izlaze iz nje.

Batch normalizacija upravo to i radi. To je vrta sloja koja je predtavljena 2015 godine i ona može adaptivno normalizovati podatke čak i kada se srednja vrijednost i varijansa mijenjaju tokom treniranja. Tokom treniranja se koristi srednja vrijednost i varijansa trenutnog batcha podataka da normalizuje uzorke, dok tokom izvođenja koristi eksponencijalni pomični prosjek srednjih vrijednosti i varijansi batcha koji je viđen tokom treniranja.
U praksi glavni efekat batch normalizacije čini se da je pomoć u propagaciji gradijenata – slično kao kod rezidualnih veza i na taj način omogućava treniranje dubljih mreža. Neke veoma duboke mreže mogu se trenirati samo ako uključuju više slojeva batch normalizacije. Na primjer batch normalizacija se često koristi u mnogim naprednim konvolucijskim mrežama koje dolaze uz Keras biblioteku kao što su: ResNet50, EfficientNet i Xception.
Primjer kako ne treba koristiti batch normalizaciju:

```
x = layers.Conv2D(32, 3, activation="relu")(x)
x = layers.BatchNormalization()(x)
```

Preporučljivo je da se aktivacija prethodnog sloja postavi nakon sloja za batch normalizaciju. Na sljedećem primjeru je prikazano kako se koristi batch normalizacija gdje je aktivacija na zadnjem mjestu:

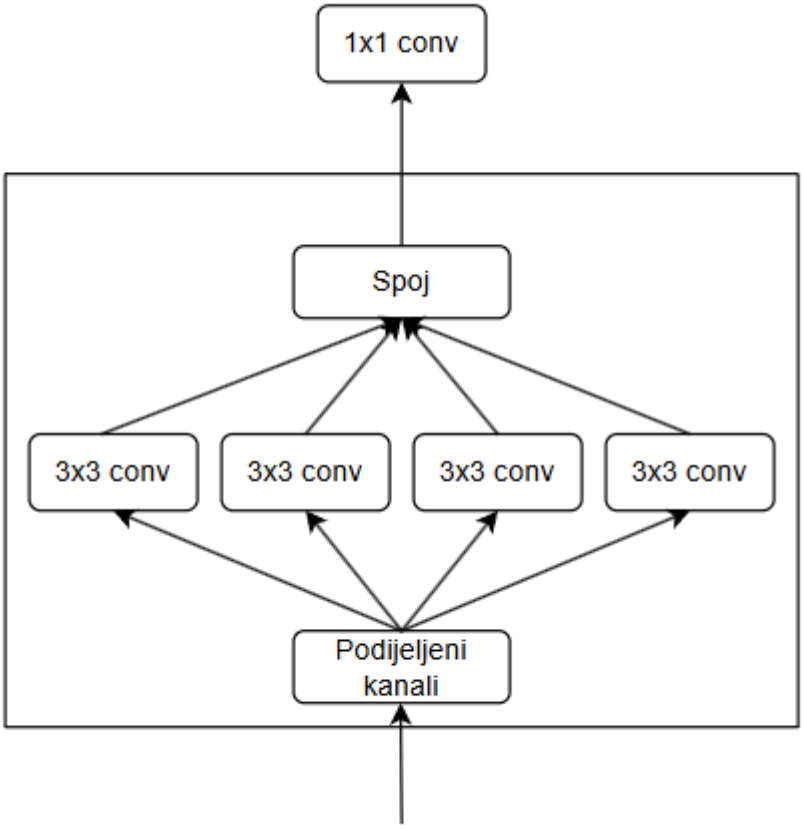


Intuitivni razlog za ovaj pristup je to što batch normalizacija centrira ulaze na nulu dok relu aktivacija koristi nulu kao granicu za čuvanje ili odbacivanje aktivnih kanala. Ako se normalizacija obavi prije aktivacije maksimalno se onda iskorištava relu funkcija. Ako se prvo primjeni konvolucija a zatim aktivacija pa tek na kraju batch normalizacija model će i dalje moći da se trenira i neće se nužno prikazivati loši rezultati.

Batch normalizacija ima mnogo specifičnosti. Jedna od glavnih specifičnosti se odnosi na fino podešavanje modela (fine-tuning). Kada se fino podesi model koji sadržava slojeve Batch normalizacije potrebno je te slojeve ostaviti „zamrznutim“ (postaviti atribut *trainable* na *False*). U suprotnom oni će nastaviti da ažuriraju svoje interne srednje vrijednosti i varijanse što može dovesti do nestabilnog ponašanja modela posebno ako se radi sa malim skupom podataka.

5. DUBINSKI SEPERABILNE KONVOLUCIJE

Postoji sloj koji se može koristiti kao zamjena za Conv2D i koji čini model manjim, lakšim i čak može dovesti do nekoliko postotaka boljih rezultata. Upravo to omogućava sloj dubinksi seperabilne konvolucije koji je u Kerasu poznat kao SeperableConv2D. Ovaj sloj izvršava prostornu konvoluciju na svakome kanalu ulaza zasebno a zatim miješa izlazne kanale pomoću tačkaste konvolucije što je u biti 1x1 konvolucija. Na sljedećoj slici je prikazano stvaranje 1x1 konvolucije.



Ovo je ekvivalentno razdvajanju učenja prostornih karakteristika i učenja karakteristika po kanalima. Na sličan način kao što se obična konvolucija oslanja na pretpostavci da obrasci u slikama nisu vezani za određene lokacije. Dubinski seperabilna konvolucija se oslanja na to da su prostorne lokacije u međurezultatima visoko korelisane dok su različiti kanali međusobno uglavnom nezavisni. Budući da je ova pretpostavka tačna za slikovne primjere koje uče duboke neuronske mreže ona također služi kako koristan uslov koji pomaže modelu da efikasnije iskoristi

svoje podatke za treniranje. Model koji ima bolje pretpostavke o strukturi informacija koji će obrađivati bit će bolji model pod uslovom da su te pretpostavke tačne.

Dubinski seperabilna konvolucija zahtjeva znatno manje parametara i uključuje znatno manje računskih operacija u poređenju sa običnom konvolucijom a pri tome ima uporedivu moć prezentacije. To rezultuje manjim modelima koji se brže konvergiraju i koji su manje skloni da model bude pretreniran (overfitting). Ove prednosti su posebno važne kada se treniraju mali modeli od početka na ograničenim skupovima podataka. Kada je riječ o modelima većeg obima, dubinski seperabilne konvolucije čine osnovu arhitekture Xception, jedne od visokoperformansnih mreža koje dolaze uz Keras biblioteku.

Uprkos mnogim zahtjevima upućenim ka NVIDIA-i dubinski seperabilne konvolucije nisu dobile ni približno isti nivo softverske i hardverske optimizacije kao obične konvolucije. Rezultat toga je da su one i dalje otprilike jednako brze kao i obične konvolucije iako koriste kvadratno manje parametara i operacija sa pokretnim zarezom. Ipak korištenje dubinski seperabilnih konvolucija i dalje je dobar primjer čak i ako ne donosi brzinu u izvođenju. Njihovo korištenje manje parametara smanjuje rizik od pretreniravanja modela a njihova pretpostavka o nekorelaciji kanala dovodi do bržeg konvergiranja modela i više otpornijim reprezentacijama.