Duboko učenje u dubinskoj analizi podataka

Dubinska analiza podataka

10. predavanje

Pripremio: izv. prof. dr. sc. Alan Jović

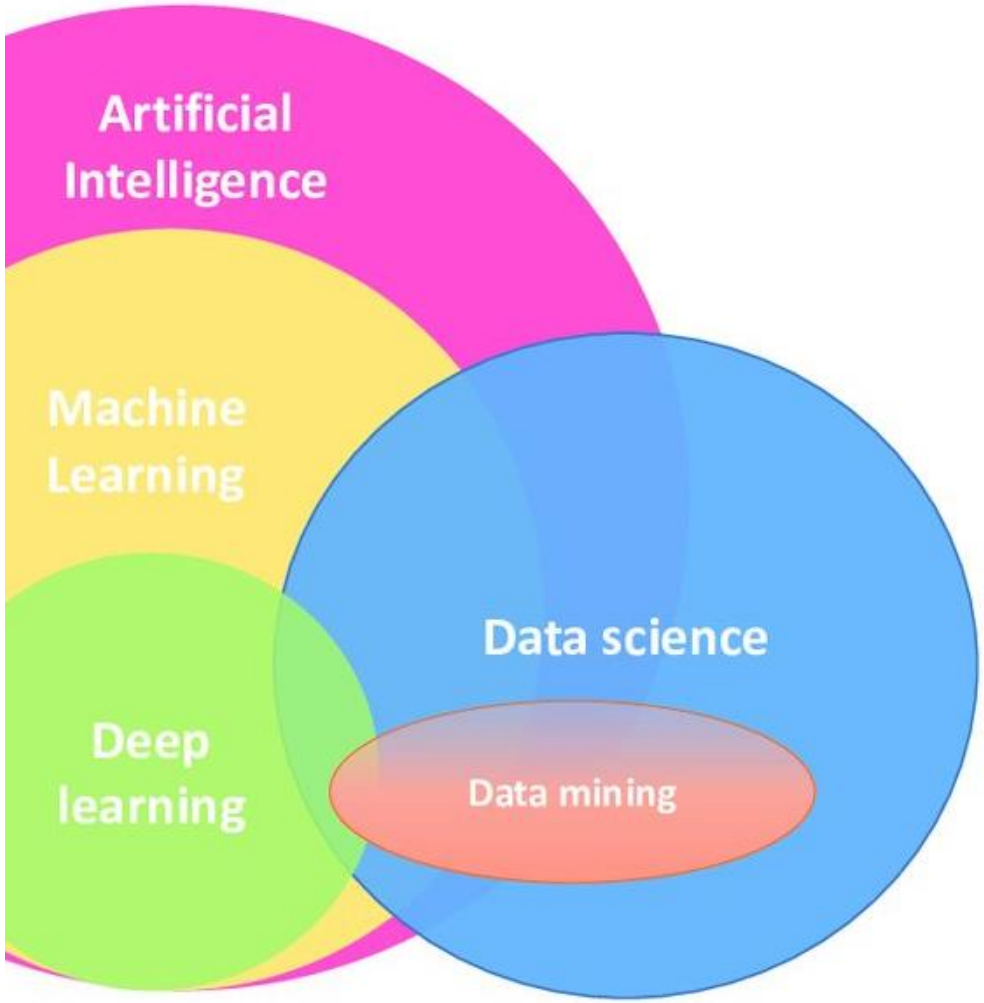
Ak. god. 2023./2024.

Sadržaj

- Koncepti i metode dubokog učenja
- Uspjeh i problemi neuronskih mreža
- Temeljne vrste dubokih neuronskih mreža
  - Konvolucijske mreže
  - Rekurentne mreže
  - Autoenkoderi
  - Transformeri

Koncepti i metode dubokog učenja

Duboko učenje

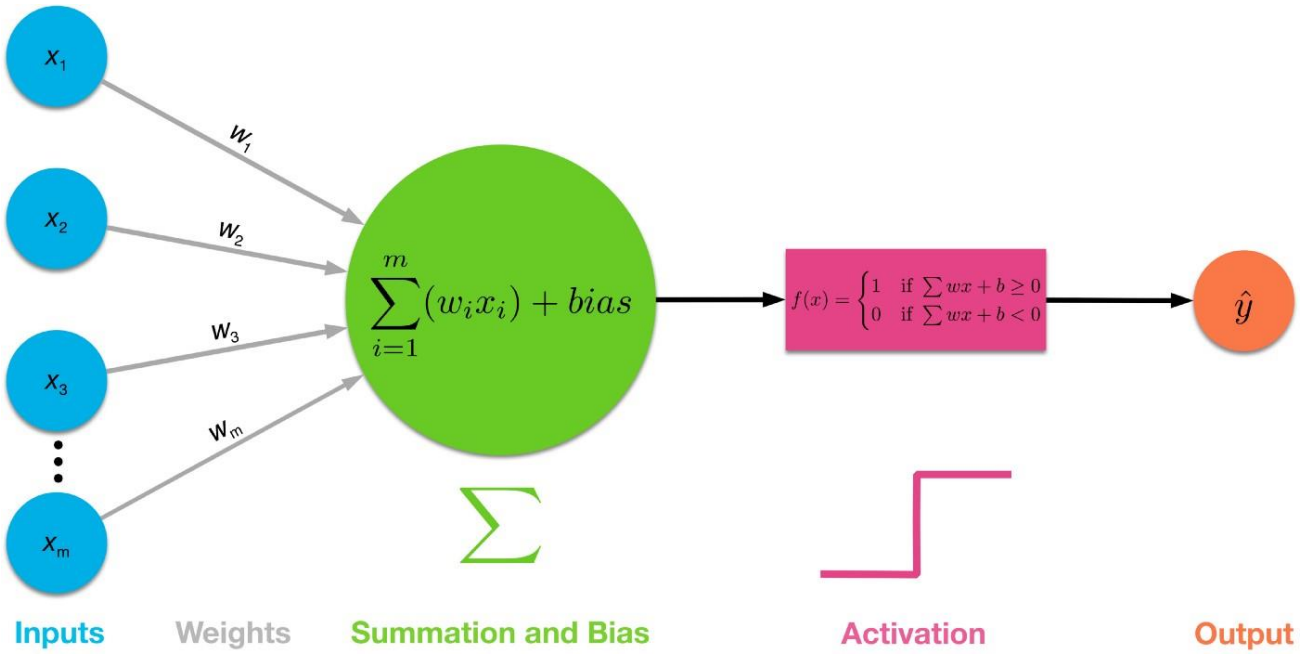


- Engl. *Deep learning*
- Područje strojnog učenja koje koristi različite arhitekture dubokih neuronskih mreža s odgovarajućim postupcima učenja
- Razvija se od 2011., s najvećim napretkom od 2016. do danas
- Nije zamjena za strojno učenje, koristi se zajedno s tradicionalnim algoritmima strojnog učenja
- Zasnovano na konceptu **učenja značajki** (engl. *feature learning*) ili **učenja reprezentacije** (engl. *representation learning*)
- Dubinska analiza podataka u specifičnim područjima koristi arhitekture ili koncepte dubokih neuronskih mreža za pronalazak korisnih obrazaca (znanje) u podacima

https://www.researchgate.net/publication/348898137_A_Survey_on_Machine_Learning-Based_Performance_Improvement_of_Wireless_Networks_PHY_MAC_and_Network_Layer/figures?lo=1

Umjetne neuronske mreže

- **Matematički model** bioloških neuronskih mreža
- Mrežu čine umjetni neuroni povezani u slojeve
  - **Jedan ulazni sloj, jedan ili više skrivenih slojeva, jedan izlazni sloj** (min. 3 sloja)
  - **Memoriju mreže čine iznosi težina veza** između neurona u slojevima
- Umjetni neuron je model koji uzima u obzir vektore težina *w* prema neuronima najčešće prethodnog sloja i određenu aktivacijsku (prijenosnu) funkciju
- Neuronske mreže mogu klasificirati podatke koji nisu linearno odvojivi



Jednoslojni perceptron

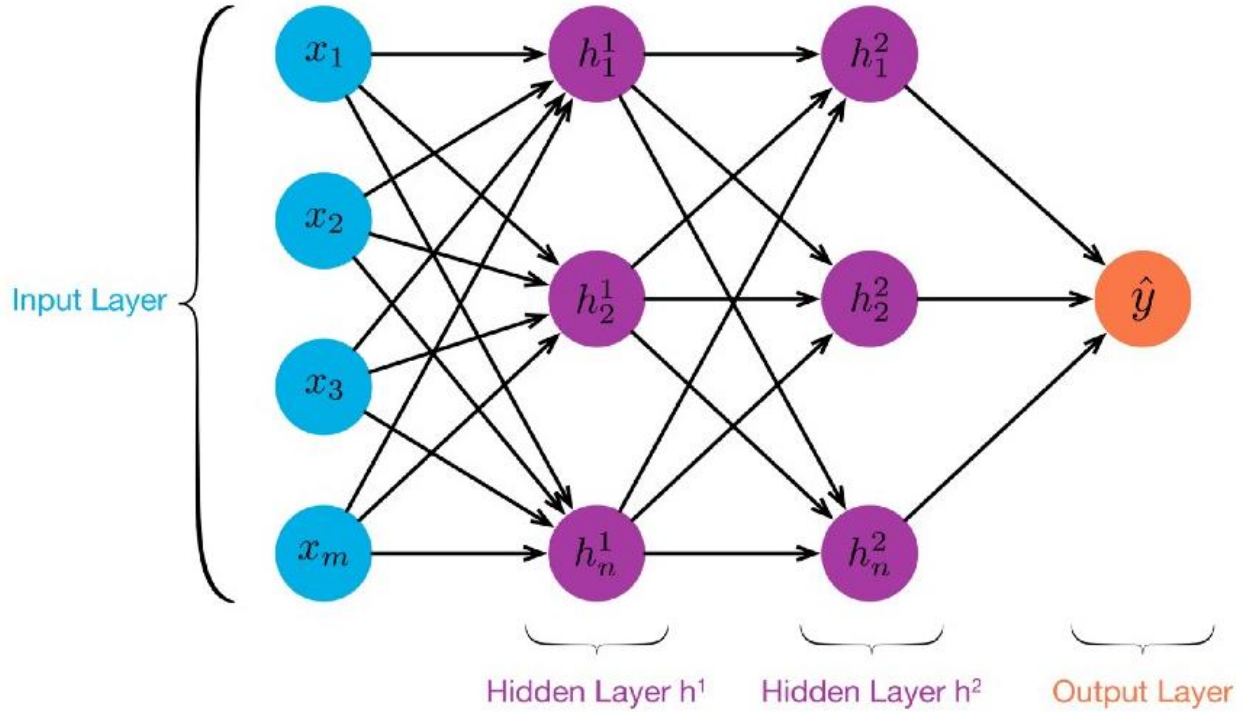
https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f

Umjetne neuronske mreže

- Umjetne neuronske mreže imaju različite varijacije:
  - **Povezanosti slojeva** (npr. višeslojni perceptron, rekurentna, konvolucijska...)
  - **Aktivacijske funkcije** (npr. sigmoidalna, linearna, *softmax*, ReLU...)
  - **Algoritama učenja i optimizatora** (npr. stohastički gradijentni spust s optimizatorom Adam)
  - **Funkcije gubitka koju treba minimizirati** (npr. srednja kvadratna pogreška, unakrsna entropija)

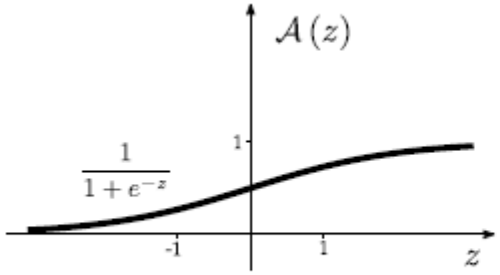
Višeslojni perceptron s unaprijednom propagacijom signala (engl. *feed-forward multilayer perceptron*) – najčešća arhitektura, svaki neuron idućeg sloja povezan sa svim neuronima prethodnog putem težina

https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f

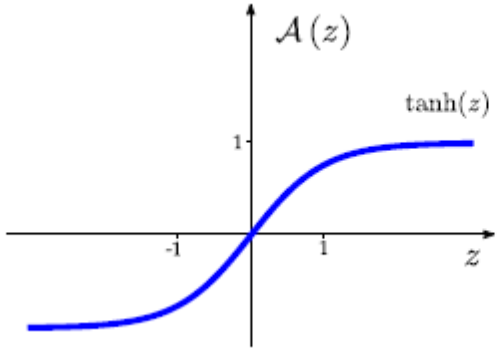


Aktivacijske funkcije

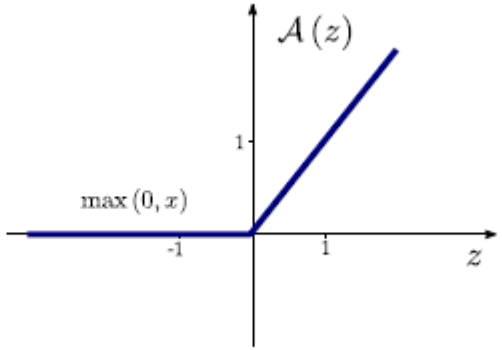
- **Aktivacijske funkcije** (engl. *activation function*, *transfer function*)
  - Razvijaju se od 1990-tih do danas
  - Najčešće rade nelinearnu transformaciju vrijednosti modela težina (suma po svim vezama umnoška težina s odgovarajućim neuronima prethodnog sloja) u nekom neuronu
  - Izbor funkcije neki put značajno utječe na uspješnost neuronske mreže
  - Većina primjenjiva na većinu vrsta slojeva neuronskih mreža



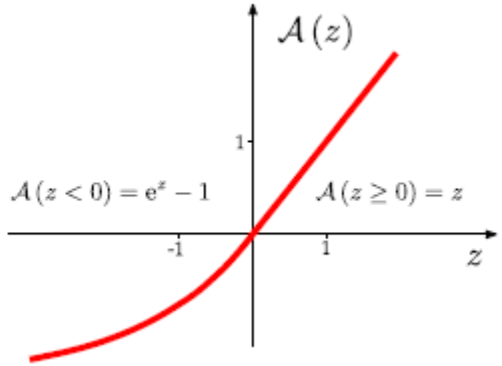
Sigmoida (logistička funkcija)



Tangens hiperbolni (**tanh**)



Funkcija **ReLU**
(*Rectified Linear Unit*)



Eksponencijalna i linearna funkcija
ELU (*Exponential Linear Unit*)

https://www.researchgate.net/publication/346898697_Thermodynamics-based_Artificial_Neural_Networks_for_constitutive_modeling

Algoritmi učenja

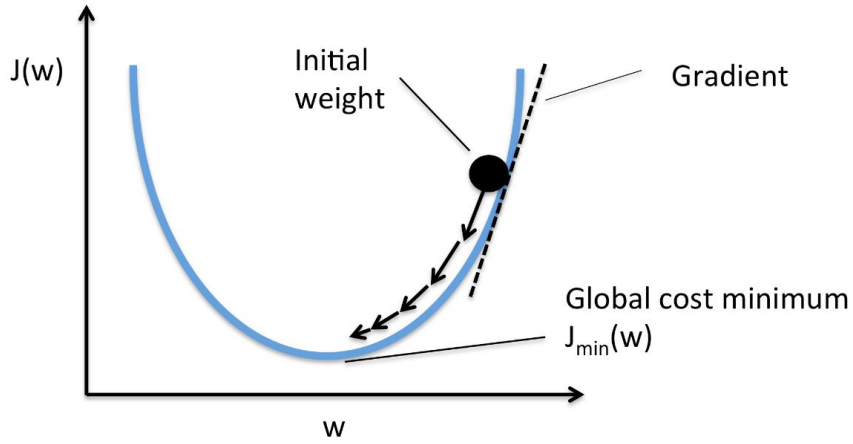
- Najšire korišteni algoritam za učenje neuronske mreže je gradijentni spust (engl. *gradient descent*)
  - **Revidira težine veza** (u određenom smjeru, tj. gradijentu) između neurona
- Za računanje gradijenta u **unutarnjim slojevima mreže** često se koristi algoritam propagacije **pogrešaka predviđanja** unazad (engl. *error backpropagation algorithm*), u odnosu na **ciljnu funkciju gubitka *J*** (engl. *loss function*), koja ima razne definicije
- **Težine veza** (kao i točnost predviđanja) se revidiraju tijekom učenja neuronske mreže kako na ulaz mreže dolaze novi primjerci za učenje
- **Jedna epoha učenja** = **jedan prolaz** (dostupnost podataka na ulazu mreže) kroz **čitav skup za učenje**, nakon jednog prolaza unaprijed revidiraju se težine svih neurona propagacijom unazad

https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/

Algoritmi učenja

- U praksi je češći **stohastični gradijentni spust** (engl. *stochastic gradient descent*, SGD)
  - Težine se revidiraju nakon svakog primjerka za učenje
  - Razlog: manji rizik za zaglaviti u lokalnom minimumu funkcije gubitka
- Danas je najčešći **gradijentni spust s podskupom primjeraka** (engl. *mini-batch gradient descent*)
  - Kompromis između jednog prolaza i stohastičnog gradijentnog skupa
  - Izabire se podskup od *k*-primjeraka nakon čega se radi revizija težina
  - Brža je konvergencija od GD-a, manje potrebnih operacija od SGD-a, može se vektorski izračunati odjednom za cijeli *batch*
  - *k* varira između 16 i 512, ovisno o zadatku i dostupnim resursima na računalu (često je 32 – 64)

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j}$$



https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/
https://sebastianraschka.com/faq/docs/gradient-optimization.html

Algoritmi učenja

- Postoje mnoga popularna proširenja (tzv. **optimizatori**) osnovnog stohastičkog gradijentnog spusta koja nastoje nekako **optimirati** proces učenja mreže
  - Cilj: brža konvergencija (manje iteracija učenja) globalnom minimumu uz zadržavanje stabilnosti učenja
  - Npr. adaptivna stopa učenja tijekom vremena, učenje momenta (dodavanje faktora prošlog gradijenta prilikom izmjene sadašnjeg)
  - Primjeri optimizatora: Momentum, Root Mean Squared Propagation (RMSProp), Adagrad, Adaptive Movement Estimation (Adam), Adam with Weight Decay (AdamW), Rectified Adam (RAdam), Madgrad, itd.
- Osim optimizatora, koristi se i tehnika predobrade podskupa primjeraka pod nazivom **normalizacija podskupa primjeraka** (engl. *batch normalization*)
  - Sloj u mreži koji djeluje tako da **standardizira** vrijednosti ulaza u taj sloj na razini cijelog *mini-batcha*
  - Učinkovita tehnika za povećanje stabilnosti učenja

https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/
https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/

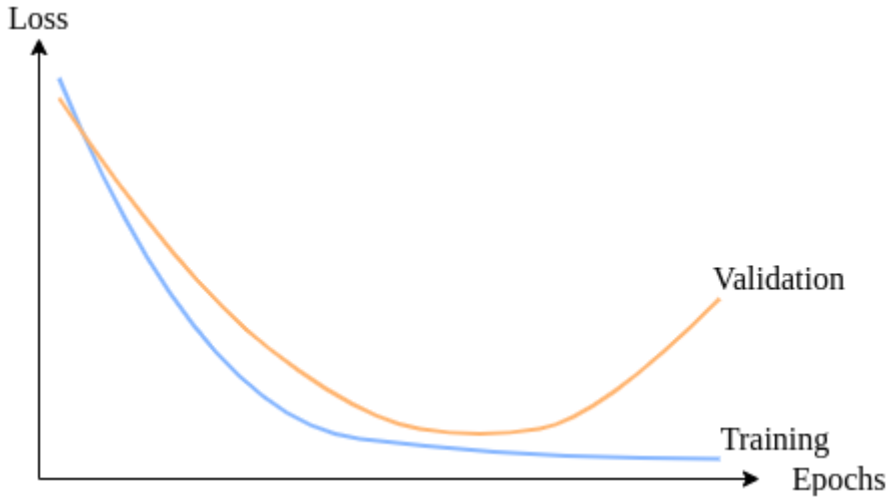
Optimizacija hiperparametara

- Neuronske mreže imaju hiperparametre koji se postavljaju prije učenja i koji se gotovo uvijek optimiraju kako bi se dobili bolji rezultati
- Optimizacija hiperparametara se radi: **slučajno**, **ručnim izborom**, **pretragom po rešetki** (engl. *grid search*, najčešće!) ili **Bayesovom optimizacijom (temeljeno na modelu)**
- **Bayesova optimizacija**
  - Odabir nekoliko slučajnih točaka u prostoru hiperparametara i evaluacija objektivne funkcije (npr. prema točnosti)
  - Rezultati se koriste za fitanje inicijalnog probabilističkog **modela** objektivne funkcije
  - Na temelju modela odabire se nove vrijednosti hiperparametara
  - Proces se ponavlja dok se ne dosegne kriterij zaustavljanja
  - Bayesova optimizacija je korisna za visokodimenzionalni prostor hiperparamatera
- Za detalje vidjeti: https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f

Optimizacija hiperparametara

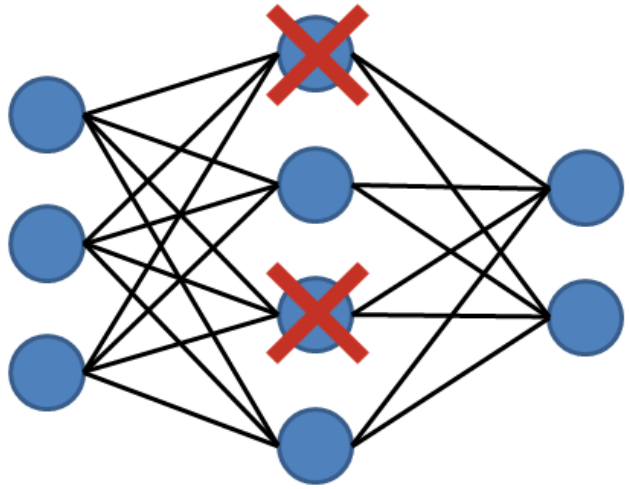
- Hiperparametri strukture neuronske mreže **(neovisno o vrsti neuronske mreže!)**
  - **Broj slojeva i broj neurona u sloju** – obično se dodaju postepeno dok pogreška mreže na validacijskom skupu pada (dobro je koristiti unakrsnu validaciju) – premala mreža = podnaučenost, prevelika mreža = prenaučenost
  - **Isključivanje neurona (engl. *dropout*)** – postotak neurona (slučajno odabranih) u pojedinom skrivenom sloju koji se ne razmatraju u jednoj iteraciji procesa učenja (prolaz unaprijed i unazadna propagacija pogreške), dobro radi za veće mreže – obično 0.1 do 0.5 (10% – 50%)
  - **Aktivacijske funkcije** – u skrivenom sloju najčešće **ReLU**, može i **tanh** ako nema puno skrivenih slojeva

Optimizacija hiperparametara



Gubitak na skupu za učenje i na validacijskom skupu

https://www.baeldung.com/cs/epoch-neural-networks



Isključivanje neurona

https://towardsdatascience.com/coding-neural-network-dropout-3095632d25ce

Optimizacija hiperparametara

- Hiperparametri procesa učenja neuronske mreže **(neovisno o vrsti neuronske mreže!)**
  - **Stopa učenja** (engl. *learning rate*) – određuje za koliko se revidiraju parametri (težine) neuronske mreže
    - Niža stopa učenja usporava proces učenja, ali mreža bolje konvergira do zadovoljavajućih rješenja
  - **Broj epoha učenja** (engl. *number of epochs*) – broj puta koliko se cijeli skup za učenje da na ulaz mreži
    - Mrežu je najbolje učiti onoliko dugo dok pogreška na validacijskom skupu pada
  - **Veličina slučajnog podskupa primjeraka** (engl. *batch size*) – broj primjeraka na ulazu mreže u **jednoj iteraciji učenja** nakon koje dolazi do **revizije težina neurona**
    - 1 epoha = broj iteracija * veličina podskupa primjeraka
    - Može biti: **cijeli skup (1 epoha = 1 iteracija)**, **jedan slučajni primjerak (SGD)** ili **slučajni podskup skupa primjeraka** (*mini batch*) (uz uzorkovanje bez ponavljanja)
  - **Hiperparametri optimizatora** učenja (različiti, ovisno o optimizatoru)

Funkcije gubitka

- **Funkcija gubitka ili sloj gubitka** (engl. *loss function*, *loss layer*) je završni sloj neuronske mreže koji specificira kako učenje penalizira razliku između predviđenog izlaza iz mreže (dobivenog aktivacijskom funkcijom) tijekom procesa nadziranog učenja i stvarnog izlaza
- Na temelju funkcije gubitka *L* računaju se gradijenti koji se koriste u algoritmu učenja mreže za reviziju vrijednosti težina među neuronima
- Funkcija gubitka ne koristi se u fazi testiranja mreže (ili u produkciji)
- Mogu se koristiti različite funkcije gubitka, ovisno o specifičnom zadatku
- Funkcije gubitka mogu se dodatno modificirati (regularizirati) za postizanje boljih rezultata

Funkcije gubitka

- Najčešće funkcije gubitka za općenite klasifikacijske i regresijske probleme:
  - Funkcija kategorijske unakrsne entropije (engl. *categorical cross-entropy, log-loss*) – za dobivanje vjerojatnosti predikcije klasa za *K* klasa, mjeri razliku između predviđene i stvarne distribucije vjerojatnosti za svaku klasu; varijanta je binarna unakrsna entropija za klasifikaciju u dvije klase

    $L = - \Sigma y_i * \log(p_i)$
  - Funkcija srednje kvadratne pogreške (engl. *mean squared error*, MSE) – za predviđanje vrijednosti kod regresijskih problema, mjeri prosječnu kvadratnu razliku u predikciji vrijednosti između predviđene i stvarne, najčešće primijenjena na izlaz linearne aktivacijske funkcije $L = (1/n) * \sum(y_i - \hat{y}_i)^2$
  - Fokalni gubitak (engl. *focal loss*) – varijacija unakrsne entropije koja se fokusira na primjerke koji se teže ispravno klasificiraju (γ) – korisno za skupove s nebalansiranim klasama, može biti dodatno utežano (α_t) s težinskim faktorima pojedinih klasa

    $L = -\alpha_t * (1 - p_t)^\gamma * \log(p_t)$

https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/

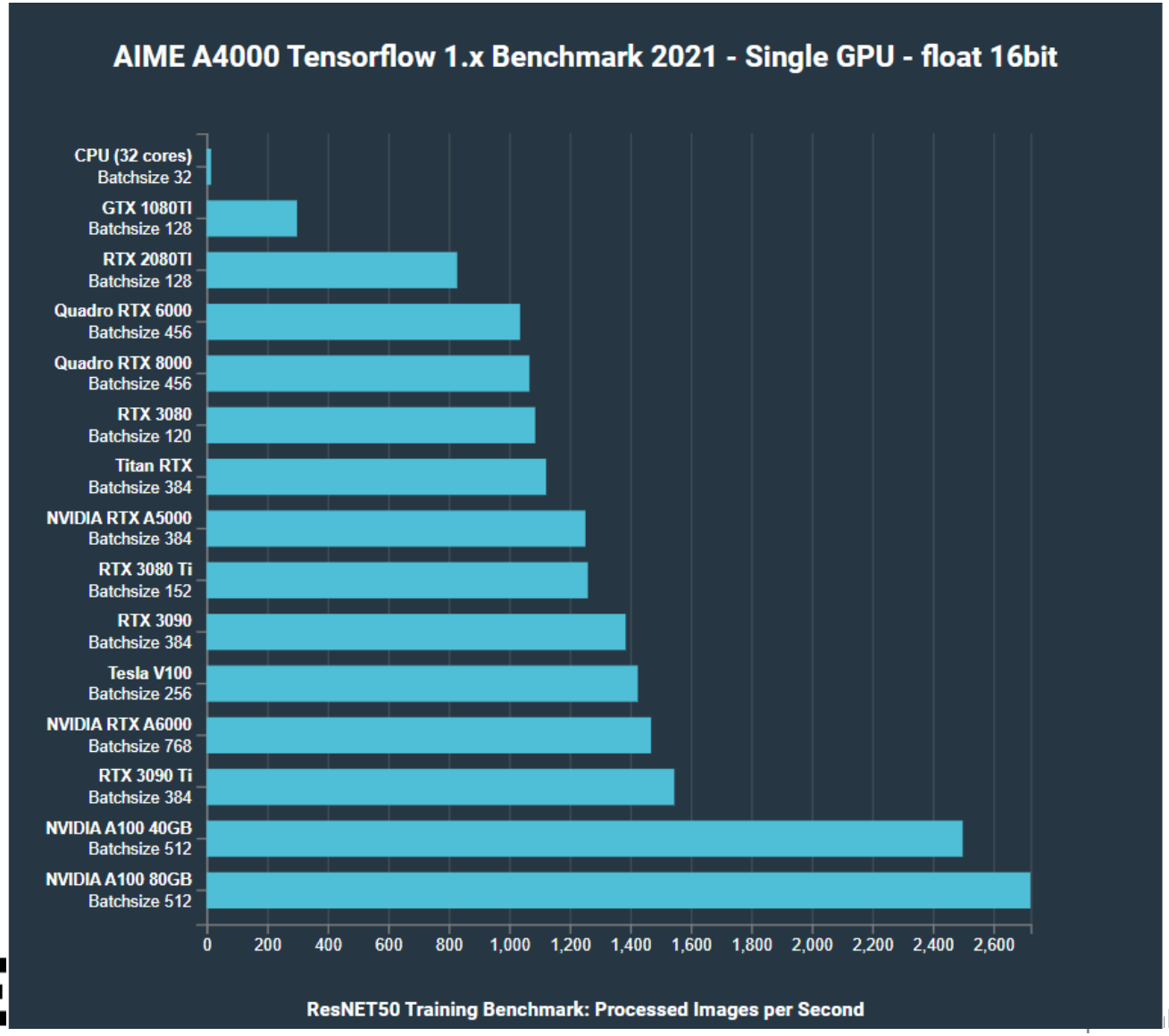
Zašto duboke neuronske mreže uspijevaju?

- Neki tehnološki razlozi:
  - Veliko **povećanje količine podataka** dostupne za učenje
    - Vidi npr. https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research
  - **Povećanje sklopovskih performansi** suvremenih računala, poglavito razvoj GPU-ova i vektorizacije učenja
- Neki arhitekturni razlozi:
  - Osmišljavanje **specijaliziranih arhitektura** za određene namjene (npr. konvolucija, rekurentni blokovi, razne podvarijante)
  - Pronalazak **novih aktivacijskih funkcija** koje omogućuju duboke modele (npr. ReLU u odnosu na sigmoidu)
  - Istraživanje **regularizacijskih metoda** (npr. *dropout* za neurone*, weight decay* za funkciju gubitka) dovodi do boljih i stabilnijih rezultata
  - **Optimizacija algoritama učenja** (npr. Adam) i **algoritama predobrade** (npr. *batch normalization*) poboljšavaju točnost i stabilnost postupaka učenja mreža

https://towardsdatascience.com/this-thing-called-weight-decay-a7cd4bcfccab

https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/

Zašto duboke neuronske mreže uspijevaju?



GPU revolucija

Configuration	float 32 training	float 16 training
CPU(32 cores)	55 days	55 days
Single RTX 2080 TI	57 hours	24 hours
Single RTX 3080	44 hours	18 hours
Single RTX 3080 TI	38 hours	17 hours
Single RTX 3090	34 hours	14.5 hours
Single RTX A6000	34 hours	14.5 hours
Single NVIDIA A100	19 hours	8 hours
4 x RTX 2080TI	16 hours	6.5 hours
4 x Tesla V100	12 hours	4 hours
4 x RTX 3090	9.5 hours	4 hours
4 x NVIDIA A100	5.5 hours	2.5 hours

- Učenje ResNet50 na 1.431.167 slika (ImageNet 2017), 50 epoha učenja na svim slikama

https://www.aime.info/en/blog/deep-learning-gpu-benchmarks-2021/

Problemi dubokih neuronskih mreža

- Neki od najznačajnijih problema dubokih neuronskih mreža su:
  - **Slaba objašnjivost modela**
    - Neuronske mreže su **crna kutija** u smislu razloga za donošenje neke odluke; iako rade dobro, uglavnom ne znamo zašto je tome tako
  - **Složenost postupka učenja**
    - Nerazumijevanje internog funkcioniranja složene neuronske mreže otežava njihovo poboljšavanje (sve se svodi na pretragu prostora hiperparametara), također zahtijevaju **značajne sklopovske resurse** da bi se dobro naučile
  - **Problem nestajućeg gradijenta**
    - Gradijenti nose informaciju u mrežama, a kada gradijent postane premalen propagacijom unazad, revizija parametara mreže (težina) postaje beznačajna, što se događa kod učenja dubokih sekvenci podataka (u RNN-u) ili pri korištenju aktivacijskih funkcija kao što je tanh za duboke potpuno povezane mreže
  - **Problem eksplodirajućeg gradijenta**
    - Gradijent naraste na velike vrijednosti (pa i do *overflowa*) zbog stalne akumulacije (multiplikacije) s faktorom većim od 1.0 u dubokim modelima ili dubokim sekvencama (npr. zbog aktivacijske funkcije kao što je tanh ili zbog inicijalno loše postavljenih težina), s rezultatom velikog gubitka ili nestabilnosti modela

Problemi dubokih neuronskih mreža

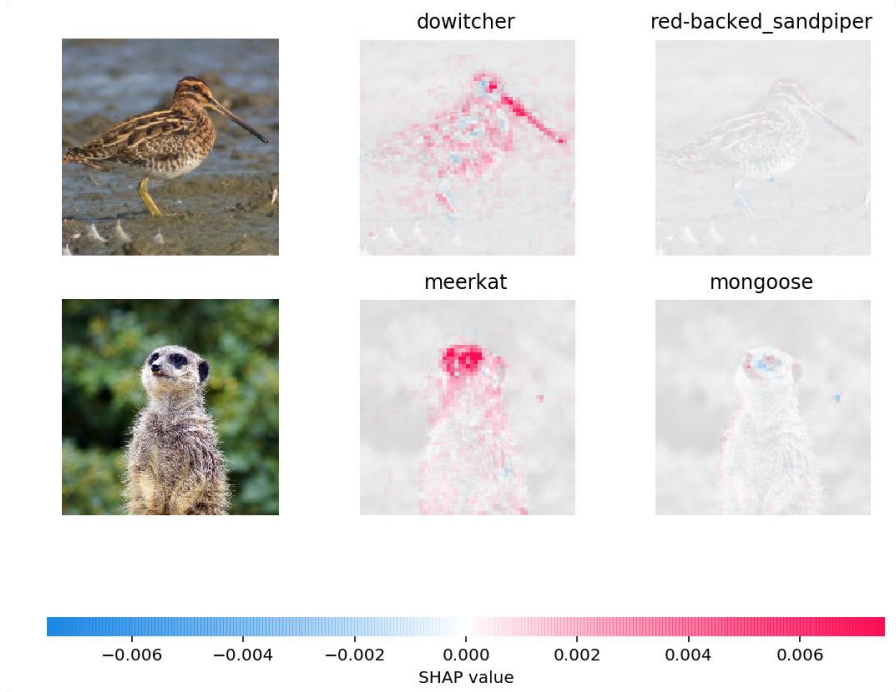
- Sklopovski resursi i poboljšavanje točnosti modela
  - Duboke mreže su moderna verzija ranijih ANN-ova s mnogo više ulaza, izlaza i međuslojeva, spojenih i organiziranih na veliki broj načina
  - Posljedica je veliki prostor hiperparametara koje treba optimirati, što je veliki trošak računalnih resursa i energije, također parametara (težina) ima često više nego podataka
  - Smanjenje pogreške modela za pola (npr. s 10% na 5%) traži otprilike **500 puta veće računalne resurse**
    - Ne može se postići efikasno slijedeći Mooreov zakon, nego zahtijeva puno procesora
    - Ekstrapolacija ovih zahtjeva dovodi do energetske i financijske neodrživosti daljnjeg poboljšanja točnosti (npr. učenje GPT-3 modela koštalo je oko 4 M$, a GPT-4 oko 100 M$ - bilijun parametara)
  - Rješenja: bolja arhitekturna rješenja, specijalizirano sklopovlje, manje mreže s optimizacijom učenja, učenje jedne mreže za više zadataka... (nema optimalnog rješenja!)

N. C. Thomson et al. Deep learning diminishing returns, IEEE Spectrum, 2021, https://spectrum.ieee.org/deep-learning-computational-cost

Problemi dubokih neuronskih mreža

- Slaba objašnjivost dubokih modela
  - Primjenjuju se post-hoc tehnike za objašnjenje nakon što je mreža već izgrađena
  - Tehnike ciljaju na objašnjavanje **kako ulaz u neki *n*-ti sloj mreže utječe na ciljnu distribuciju modela po klasama**
    - Objašnjenje je najčešće vizualnog tipa – vizualizira se istaknutost značajki u određenom sloju
    - Primjeri: SHAP Explainer, Gradient Explainer

Primjer: SHAP Explainer za dvije klase s najvećom vjerojatnosti u 7. sloju neuronske mreže – crveni pikseli povećavaju vjerojatnost pojave određene klase, a plavi smanjuju
https://github.com/slundberg/shap



Problemi dubokih neuronskih mreža

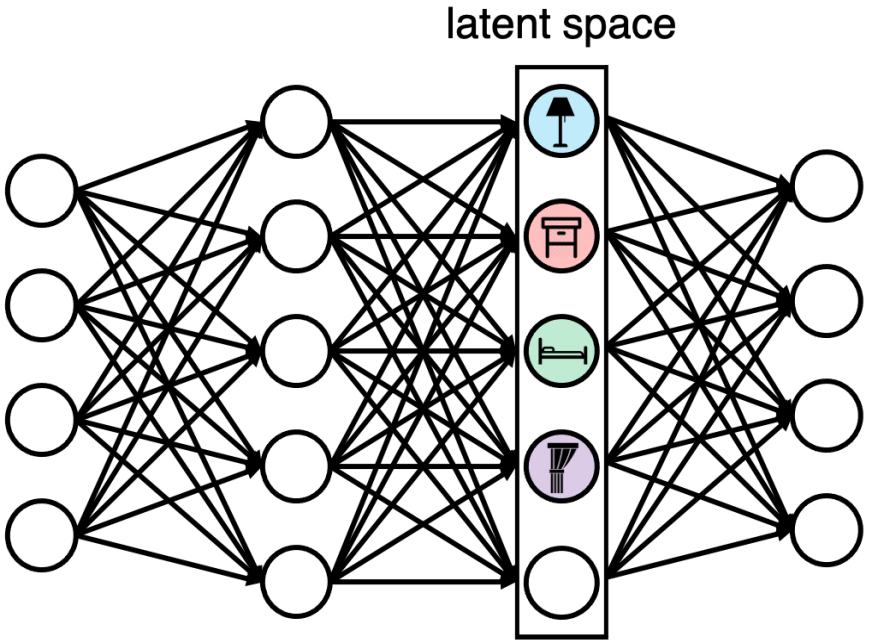
- Slaba objašnjivost dubokih modela
  - Post-hoc tehnike se više koriste za „debugiranje” neuronske mreže (interpretabilnost na razini razvojnog inženjera), a mnogo manje za transparentnu komunikaciju s korisnicima
  - Ključni pojam za bolju objašnjivost: **razmrsivanje ili otpetljavanje** (engl. ***disentanglement***) neuronske mreže
    - Proučavanje načina na koji se informacija propagira kroz neuronsku mrežu
    - Idealni slučaj razmršenosti: samo jedan neuron u određenom sloju može se izravno povezati s nekim konceptom (objektom, klasom, ciljnom vrijednosti)
    - Cilj je da svaki neuron u dijeliću mreže ima razumljivu interpretaciju

Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, Chudi Zhong "Interpretable machine learning: Fundamental principles and 10 grand challenges," Statist. Surv. 16, 1-85, (2022)

Bhatt, U., Xiang, A., Sharma, S., Weller, A., Taly, A., Jia, Y., Ghosh, J., Puri, R., Moura, J. M. F. and Eckersley, P. (2020). Explainable Machine Learning in Deployment. In Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency 648–657. Association for Computing Machinery

Problemi dubokih neuronskih mreža

- Najčešći slučaj – informacija o konceptu raspršena na više neurona u mreži, također, više skroz različitih koncepata aktivira isti pojedinačni neuron u nekom sloju (nalazi se na istoj osi latentnog prostora sloja)
- Noviji pristupi rade na boljoj objašnjivosti koncepata (boljem razmrsivanju) tako da se **dodaju ograničenja na latentni prostor koja dekoreliraju koncepte** (slično PCA)
- Ograničenja se uče iz dodatnog skupa podataka o konceptima nakon što je izvorna mreža već izgrađena, što ne škodi točnosti, ali poboljšava objašnjivost

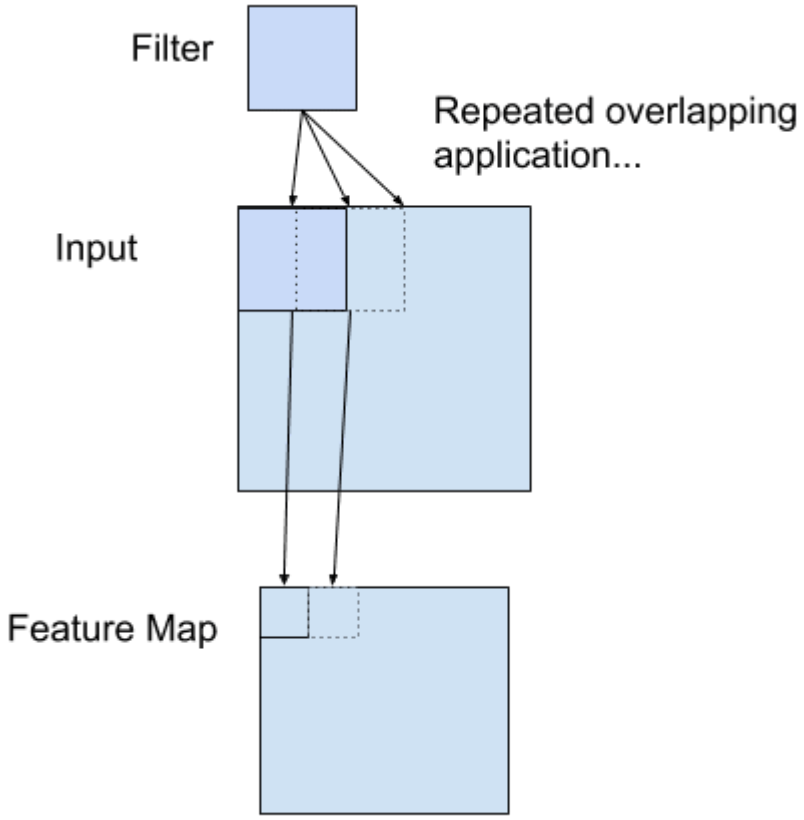


Chen, Z., Bei, Y. and Rudin, C. (2020). Concept whitening for interpretable image recognition. Nature Machine Intelligence 2 772-782.

Vrste umjetnih neuronskih mreža

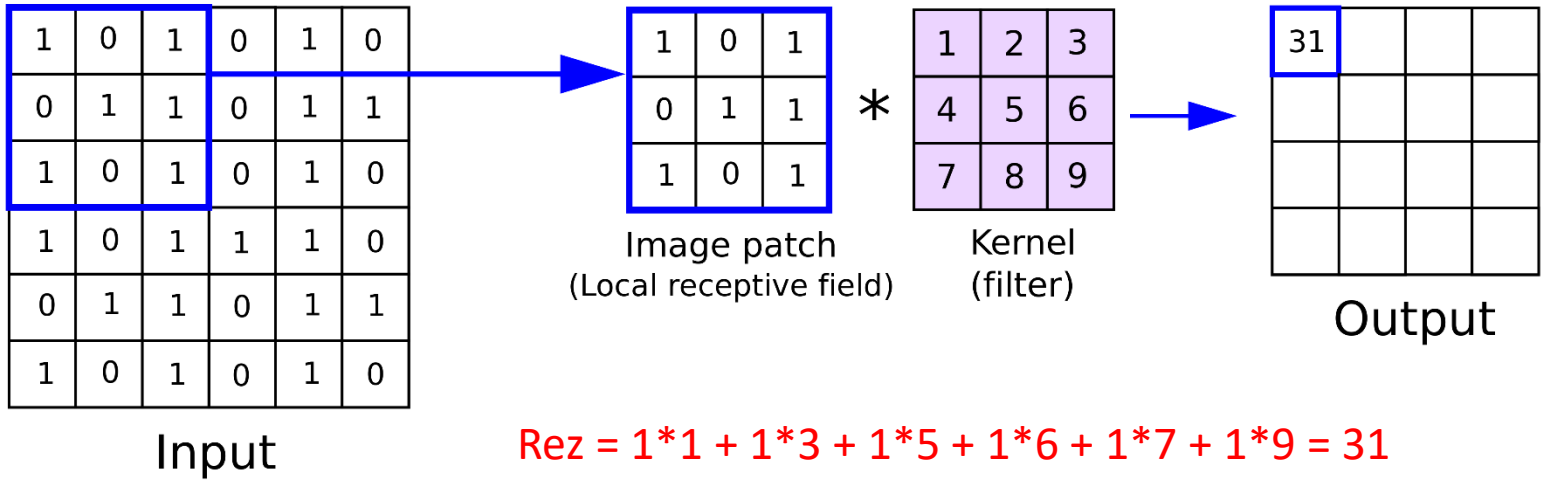
Konvolucijska mreža

- **Konvolucijski sloj** (engl. *convolution layer*)
  - Svaki neuron računa **konvoluciju** s nekim neuronima prethodnog sloja – tzv. receptivnim poljem tog neurona (engl. *receptive field*)
    - Konvolucija je **linearna operacija** koja množi skup težina u obliku matrice x * x (npr. 3*3), tzv. jezgra (engl. *kernel*) ili filtar, s receptivnim poljem (niz ili matrica y * y) u vidu **skalarnog produkta** te rezultira u jednom broju (skalaru)
    - Na rezultat (tzv. *feature map*) konvolucije filtra s receptivnim poljem primijenjuje se neka aktivacijska funkcija, npr. ReLU
  - Filtar se primijenjuje **sustavno** na svaki komadić ulaznog podatka, slijeva nadesno ili odozgo prema dolje, sa ili bez preklapanja s prethodnim receptivnim poljem (uz korak za *n* mjesta, engl. *stride*)



https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

Konvolucija



https://anhreynolds.com/blogs/cnn.html

W = dubina ulaza
K = veličina jezgre
P = veličina dopune s nulama na rubovima slike (engl. *padding*)
S = veličina koraka

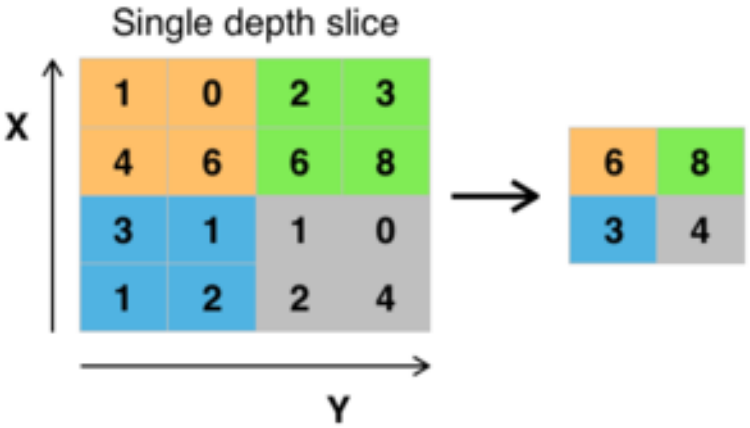
Rez = 1*1 + 1*3 + 1*5 + 1*6 + 1*7 + 1*9 = 31

- **Primjer:** ulaz: 6 x 6, receptivno polje neurona dimenzije 3 x 3, filtar dimenzija 3 x 3, izlaz je dimenzije 4 x 4. Receptivno polje se u ovom primjeru pomiče za 1 korak udesno i prema dolje, u ovom slučaju dobili smo na izlazu sliku smanjene dimenzije
- Općenita formula za broj neurona sloja izlaza je $O = (W - K + 2*P) / S + 1$ = (u primjeru) $(6 - 3 + 2*0) / 1 + 1 = 4$
- **Glavni zadatak konvolucijske mreže**: naučiti vrijednosti težina filtra (obično se počinje sa slučajnim vrijednostima)

Konvolucijska mreža

https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Max_pooling.png

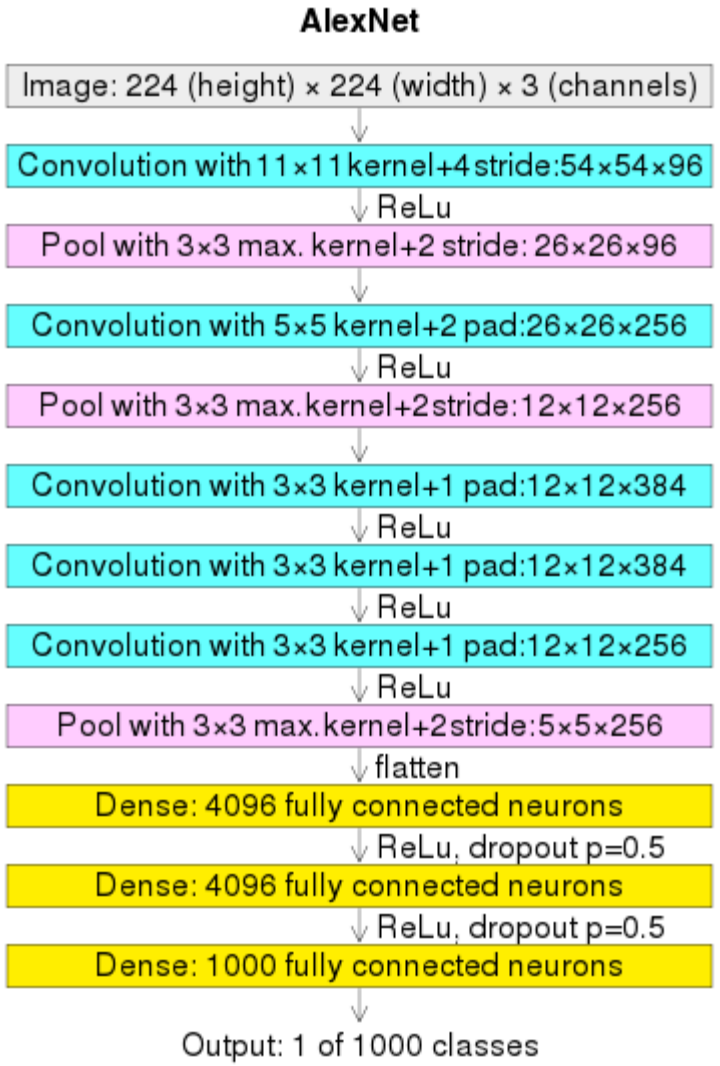
- Konvolucijska mreža **smanjuje broj parametara** koji se trebaju naučiti u odnosu na višeslojni perceptron
  - razmatra samo receptivno polje za pojedinačni neuron idućeg sloja (a ne sve neurone prethodnog sloja)
- Daljnja redukcija dimenzionalnosti reprezentacije unutar konvolucijske mreže postiže se uvođenjem sloja sažimanja (engl. *pooling layer*)
  - Poduzorkovanje na razini dijelova izlaza iz prethodnog sloja
  - Obično se koristi tehnika najveće vrijednosti (engl. *max pooling*) ili srednje vrijednosti (engl. *average pooling*)
- Sloj sažimanja, osim redukcije, dovodi i do boljeg učenja značajki sve višeg nivoa kako slojevi duboke mreže napreduju



- Primjer sažimanja s veličinom sloja sažimanja 2x2 u odnosu na izlaz prethodnog sloja, tehnika sažimanja najvećom vrijednosti

Konvolucijska mreža

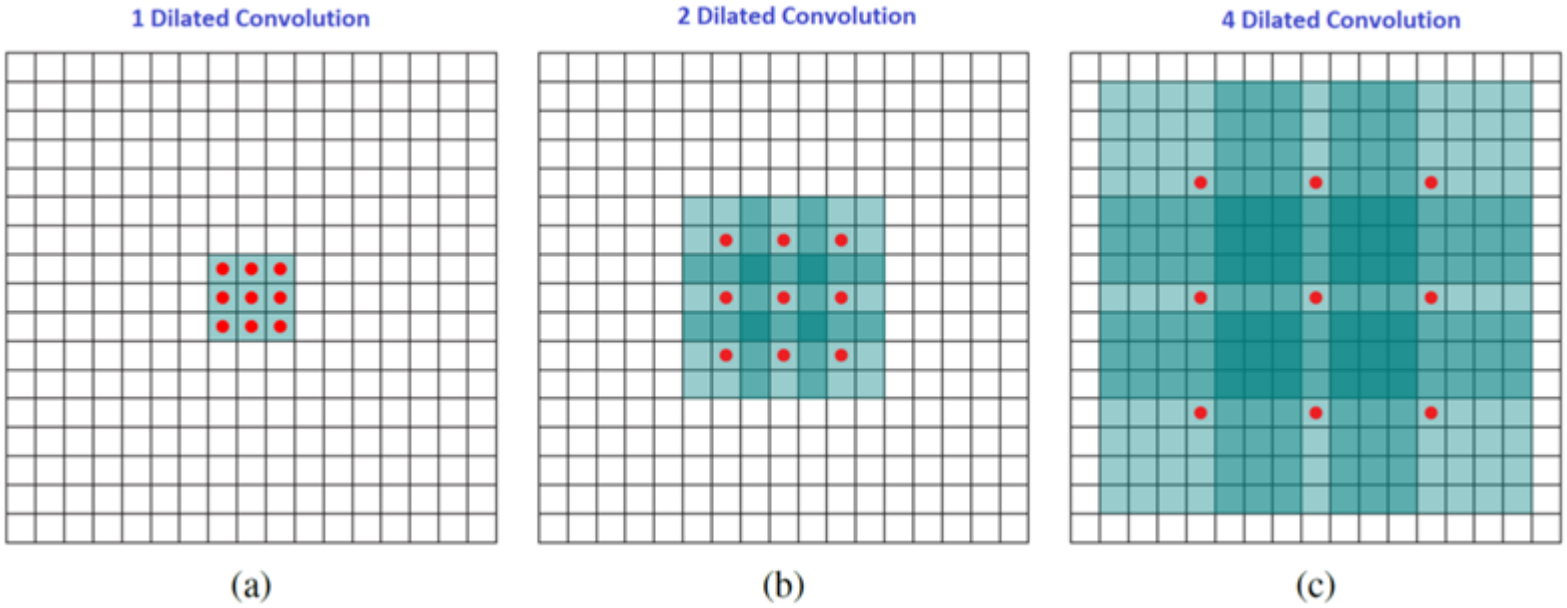
- Konvolucijska mreža često sadrži i **gusto povezane slojeve na reduciranom skupu neurona** dublje u mreži nakon operacije izravnavanja (engl. *flatten*) zadnjeg sloja sažimanja
  - Značajke dobivene konvolucijom kombiniramo nelinearnim transformacijama višeslojnog perceptrona radi bolje klasifikacije/predikcije
- Složenije neuronske mreže primijenjuju **više različitih filtara** nad ulaznom slikom (koja obično ima 1 ili 3 kanala), što je definirano u arhitekturi određenom s brojem kanala *y*: $z \times z \times y$
- Konvolucija je ustvari višedimenzijski tenzor $f \times f \times c \times c'$, gdje je *f* dimenzija filtra, *c* je broj ulaznih kanala, a *c'* broj različitih filtara koje primijenjujemo nad ulaznim kanalima, izlaz ima *c'* kanala
- Nadesno: primjer arhitekture mreže AlexNet za klasifikaciju objekata iz slika, 1000 klasa, preko 1 M slika, 60 milijuna parametara



https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Comparison_image_neural_networks.svg

Konvolucijska mreža

- Ponekad je dobro imati veće receptivno polje a zadržati istu računsku efikasnost
- Rješenje: dilatacija jezgre
- Pikseli koji se u tom slučaju uzimaju u obzir:



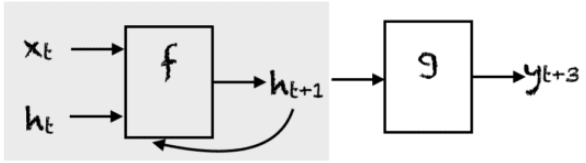
https://towardsdatascience.com/understanding-2d-dilated-convolution-operation-with-examples-in-numpy-and-tensorflow-with-d376b3972b25

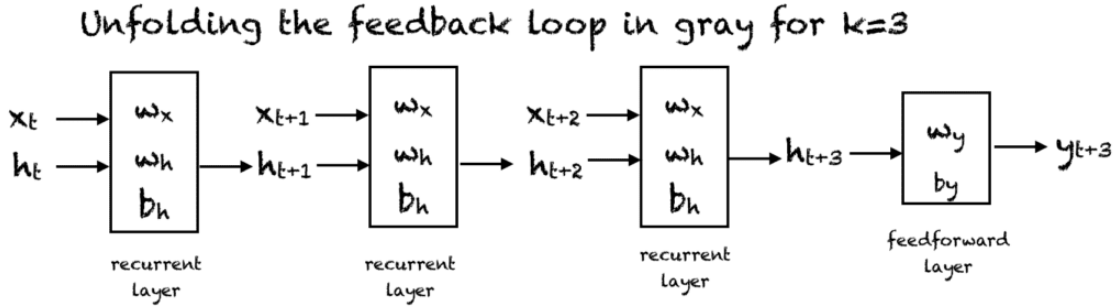
Konvolucijska mreža – hiperparametri

- Raspored slojeva konvolucijske mreže (arhitektura) najčešće se predlaže ovisno o zadatku na temelju literature i **rijetko se optimira**
- Najčešći **hiperparametri** koje treba **optimirati**:
  - **veličina jezgre** (obično 2x2, 3x3, do 5x5), neki put različite dimenzije i u istom sloju (*Inception* moduli)
  - **korak** (obično 1, 2 ili 3, rjeđe veći)
  - **veličina dopune** (od 0 do veličine jezgre – 1, često 0, rjeđe ostalo)
  - **dilatacija jezgre** – po *defaultu* = 1, ako se žele preskočiti neki pikseli, postavlja se na 2 (tada se jezgra širi na 7x7) ili na više
  - **broj različitih filtara** u svakom sloju (radi fokusa na detalje, dublji slojevi imaju više filtara i manju dimenziju)
  - **veličina sloja sažimanja** (obično od 2x2 do 4x4, veća redukcija dimenzije dovodi do gubitka informacije)
  - **vrsta sloja sažimanja** (najčešće najvećom vrijednosti, rjeđe prosječnom ili nešto treće)

Rekurentne neuronske mreže

- Višeslojni perceptron i konvolucijske mreže ne mogu učinkovito rukovati sa slijednim podacima (vremenski niz, tekst), jer razmatraju samo trenutni ulaz i ne pamte prijašnje ulaze
- Rješenje su **povratne** (**rekurentne) mreže**
- **Rekurentni sloj** (engl. *recurrent layer*)
  - Izlaz neuronske mreže ovisi o nizu aktivacija ulaznih vrijednosti koje imaju određeni poredak (u vremenu ili prostoru)
  - Rekurentni blok prima vrijednost x_t i izlaz (tzv. skriveno stanje) prethodnog bloka h_t te množeći ih odgovarajućim težinama i koristeći aktivacijsku funkciju daje novi izlaz neurona sljedećem bloku
  - Blokova u jednom horizontalnom sloju može biti više (npr. 20, ako želimo učiti na vektoru vrijednosti duljine 20)





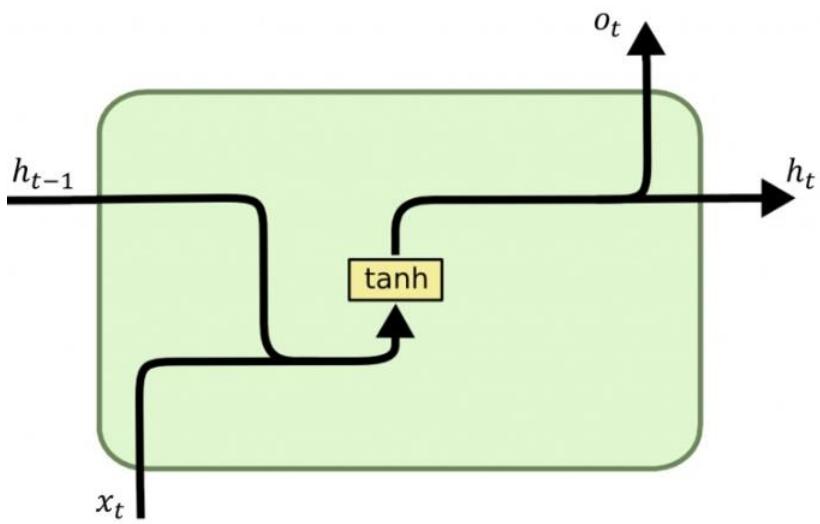
https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/

Rekurentne neuronske mreže

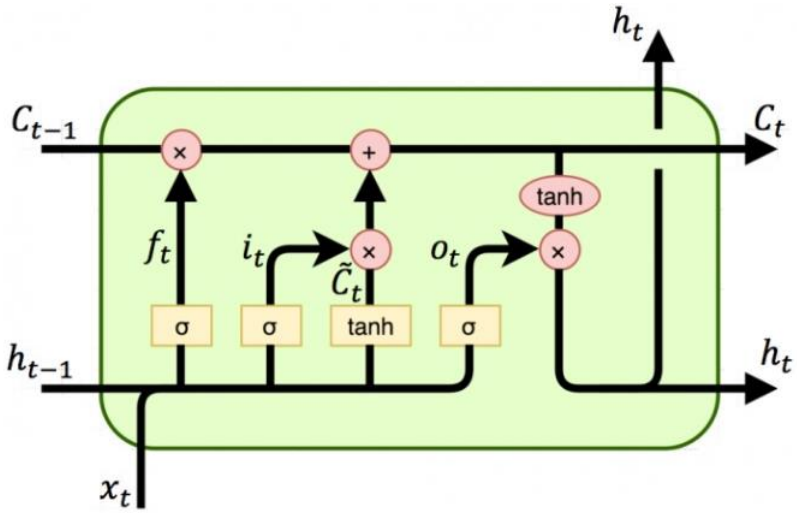
- Težine w_x i w_h su dijeljene između blokova (svi imaju iste težine), tijekom propagacije pogreške koriste se za izmjenu gradijenata
- Izvorne rekurentne mreže imaju neke **nedostatke:**
  - **Problem nestajućeg gradijenta i eksplodirajućeg gradijenta** kod učenja dugih sekvenci – propagacija pogreške unazad može kod duljih sekvenci (više izmjena unazad kroz vrijeme) dovesti do nestajanja izmjene težina i nemogućnosti učenja (kao posljedica matričnog množenja)
  - **Nemogućnost uočavanja dugoročnih ovisnosti** u podacima – RNN-ovi su osmišljeni tako da dobro uočavaju ovisnosti među slijednim podacima tako da propagiraju informaciju sljedećoj ćeliji, ali ako postoji dugoročna ovisnost onda se ona teže uočava
  - **Problemi s obradom sekvenci varijabilne duljine** – broj blokova je fiksan, što otežava obradu podataka kod kojih je informacija sadržana u sekvencama varijabilne duljine (kao što je tekst)
- Ove nedostatke u većoj mjeri pokrivaju novije varijante rekurentnih slojeva (blokova) mreže
- Razlika između vrsta rekurentnih blokova je u **složenosti (funkcionalnosti) bloka**

Rekurentne neuronske mreže

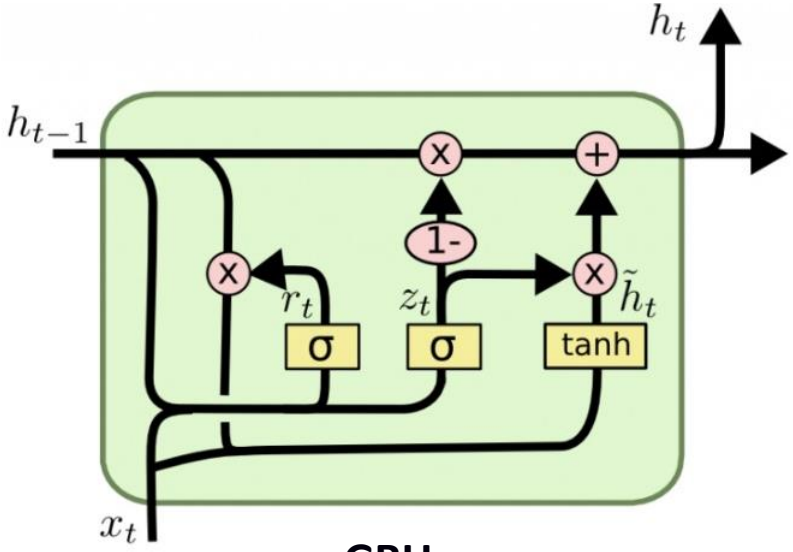
- **Tri su najčešće vrste rekurentnih blokova**:
  - Izvorni rekurentni blok (blok RNN) – najjednostavniji, prednost je brzina učenja
  - LSTM (engl. *Long Short-Term Memory*) – dobar za učenje dugotrajnih ovisnosti na većim skupovima podataka
  - GRU (engl. *Gated Recurrent Unit*) – brz i točan na kratkotrajnijim sekvencama podataka



RNN



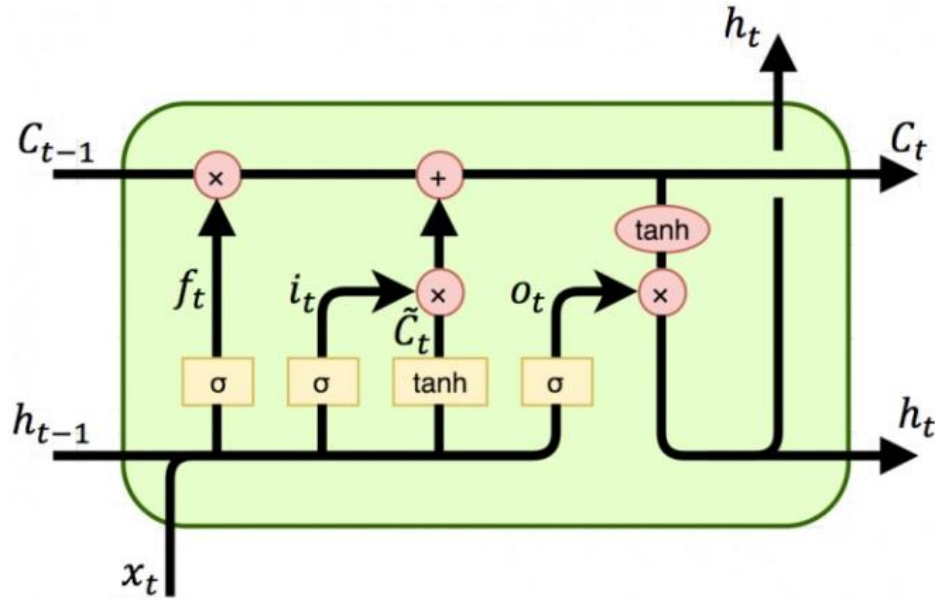
LSTM



GRU

Rekurentne mreže – LSTM

- **Blok LSTM** sastoji se od tri glavna dijela (podsloja):
  - **Vrata zaboravljanja** (engl. *forget gate*) – f_t
    - Razmatra prethodno skriveno stanje h_{t-1} i trenutni ulaz x_t i odlučuje koji dio informacije iz prethodnog stanja treba maknuti, jer nije relevantna za trenutačni ulaz
  - **Vrata novog ulaza** (engl. *input gate*) – dva dijela: i_t i $\tilde{C}_t$
    - Odlučuje koji dio informacije treba propustiti na temelju ulaza i prošlog skrivenog stanja ćelije h_{t-1}
    - Kombinira izlaz iz sigmoide i funkcije tanh za dobivanje stanja C_t
  - **Izlazna vrata** (engl. *output gate*) – o_t
    - Selektivno ispušta informaciju o novom skrivenom stanju h_t, koje ovisi prošlome skrivenom stanju h_{t-1} i o trenutačnom ulazu x_t, kao i o trenutačnom stanju C_t
    - h_t ujedno predstavlja izlaz (output) bloka
    - C_t je stanje bloka i služi za dugoročno pamćenje



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

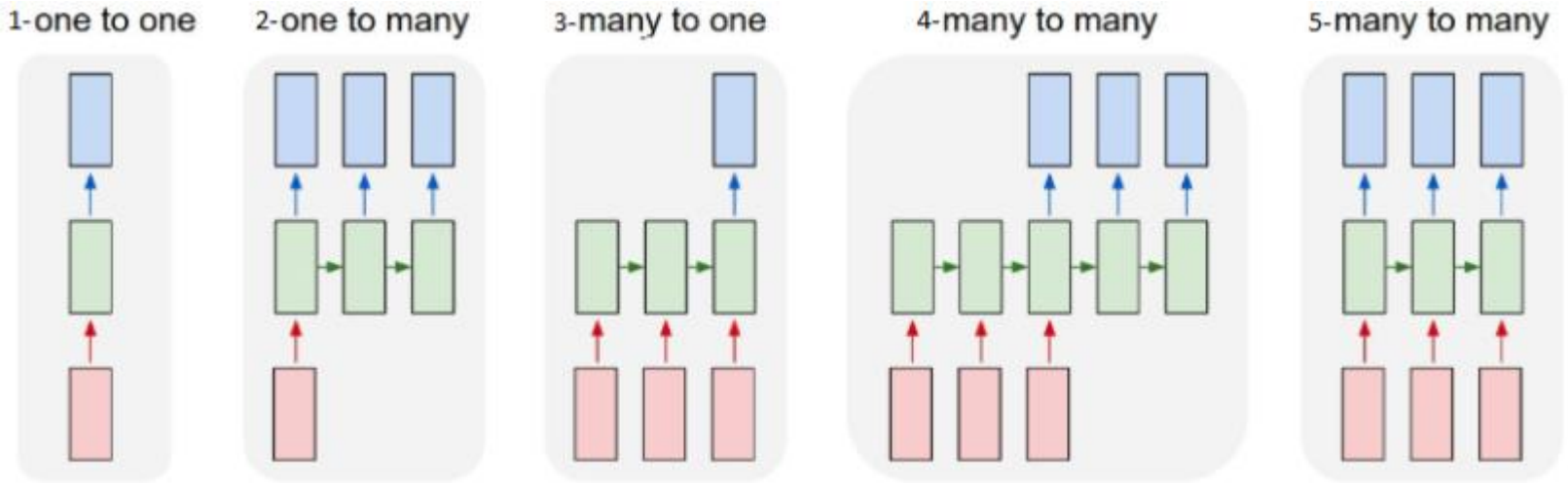
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

h_t , C_t : hidden layer vectors.
x_t : input vector.
b_f , b_i , b_c , b_o : bias vector.
W_f , W_i , W_c , W_o : parameter matrices.
σ , tanh : activation functions.

Rekurentne mreže – slaganje blokova u slojeve

- Rekurentni blokovi slažu se u rekurentne mreže na različite načine, ovisno o primjeni:



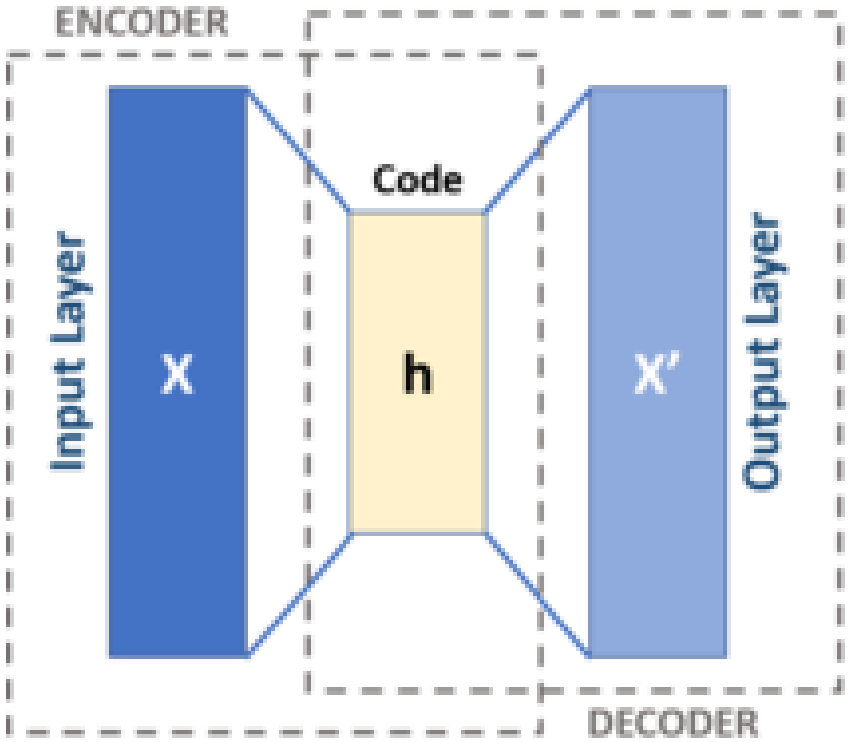
- Crveno su označene ulazne vrijednosti *x*, zeleno rekurentni blokovi (RNN, GRU, LSTM) s aktivacijom *h*, a plavo izlazne vrijednosti *y* ili *o*

http://dprogrammer.org/rnn-lstm-gru#more-1668

- Osnovna **varijanta 1 na 1** (engl. *one-to-one*) je jednostavna neuronska mreža s nelinearnom transformacijom
- **Varijanta 1 na *N*** (engl. *one-to-many*) obično se koristi za generiranje tekstnog opisa nekog ulaza, npr. opis slike
- **Varijanta *N* na 1** (engl. *many-to-one*) obično se koristi za klasifikaciju, npr. analiza sentimenta ili klasifikacija vremenskog niza
- **Varijanta *N* na *N*** (engl. *many-to-many*) obično se koristi za prevođenje teksta s jednog jezika na drugi
- **Slojeva s rekurentnim blokovima** (skrivenih slojeva) **često bude više**, neovisno o vrsti bloka i načinu slaganja blokova u slojeve
  - Više slojeva doprinosi hijerarhijskoj reprezentaciji kompleksnih problema, svaki idući sloj uzima na ulazu skup skrivenih stanja blokova prethodnog sloja

Autoenkoderi

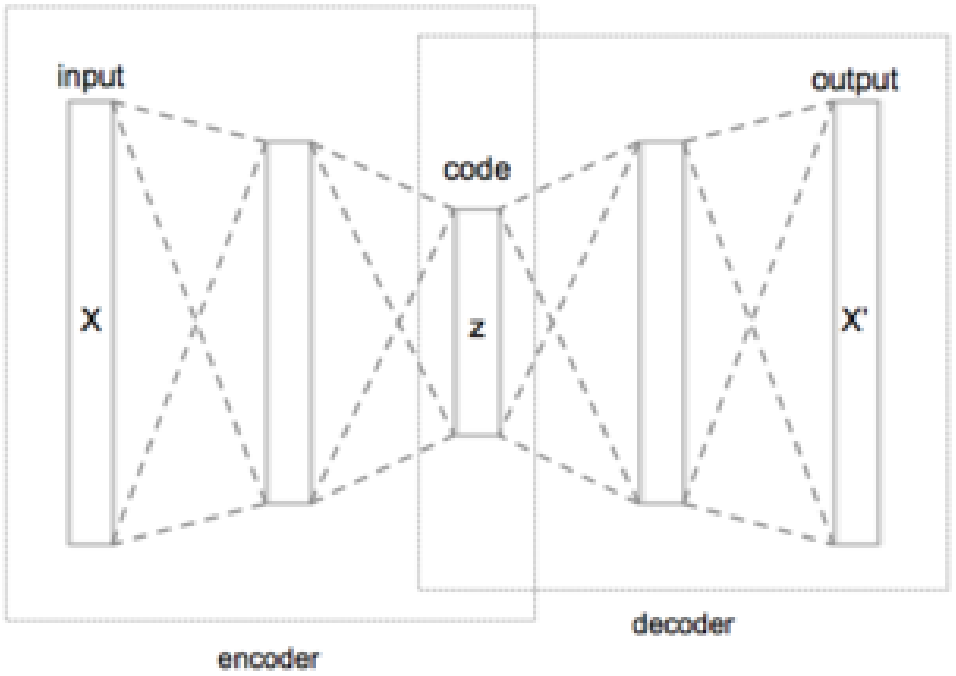
- Neuronska mreža koja se često koristi za **nenadzirano učenje,** a sastoji se od dva dijela: kodirajućeg dijela (koder, engl. *encoder*) i dekodirajućeg dijela (dekoder, engl. *decoder*)
- **Glavni zadatak kodera**: naučiti unutarnju niskodimenzionalnu reprezentaciju (naučiti značajke) ulaznog sloja, tzv. kôd (engl. *code*) ili usko grlo (engl. *bottleneck*)
- **Glavni zadatak dekodera:** transformirati kôd u vjerno rekonstruiranu verziju ulaznih podataka
- Kôd uvijek samo aproksimira ulazni sloj, izvlači najbitnije **značajke**
- Kao i kod ostalih mreža, kôd (skriveni sloj) je **nelinearna transformacija** ulaznih vrijednosti
- Primjene: smanjenje dimenzionalnosti, kompresija signala (ili slike), uklanjanje šuma, izlučivanje značajki, generiranje signala (ili slike), preporučivanje sadržaja...



https://en.wikipedia.org/wiki/Autoencoder#/media/File:Autoencoder_schema.png

Autoenkoderi

- U najjednostavnijoj varijanti, **jedan unutarnji sloj** predstavlja kôd koji je gusto povezan s ulaznim slojem i rezultat je aktivacijske funkcije (npr. sigmoida, ReLU) nad sumom umnoška težina i ulaznih neurona
- Autoenkoderi mogu imati **više skrivenih slojeva** i u kodirajućem i u dekodirajućem dijelu, s jednim najmanjim unutarnjim slojem – kodom, čime mogu učiti **kompleksnije funkcije** preslikavanja
- Autoenkoder može biti i samo unutarnji (ili početni) dio složenije duboke neuronske mreže
- Koder i dekoder mogu biti simetrični ili asimetrični



https://en.wikipedia.org/wiki/Autoencoder#/media/File:Autoencoder_structure.png

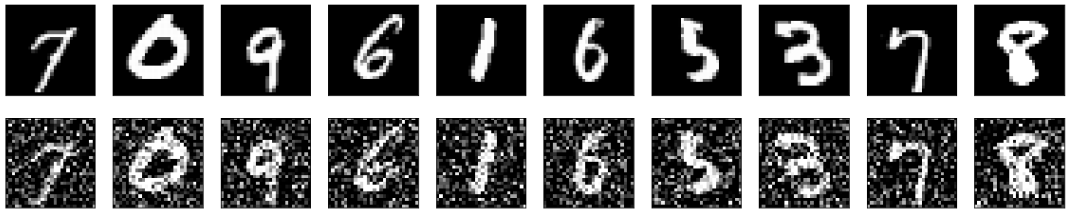
Autoenkoderi – varijante

- Autoenkoderi se mogu koristiti na različite načine za različite ciljeve, stoga mogu značajno varirati u svojoj arhitekturi
- Najčešće varijante:
  - **Rijetki autoenkoderi** (engl. *sparse autoencoder*)
  - **Autoenkoderi za uklanjanje šuma** (engl. *denoising autoencoder,* DAE)
  - **Varijacijski autoenkoderi** (engl. *variational autoencoder,* VAE)
  - **Naslagani autoenkoderi** (engl. *stacked autoencoder,* SAE)
  - Hibridni pristupi (npr. naslagani rijetki autoencoder - SSAE, varijacijski autoenkoder za uklanjanje šuma - DVAE, itd)

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Autoenkoderi – autoenkoderi za uklanjanje šuma

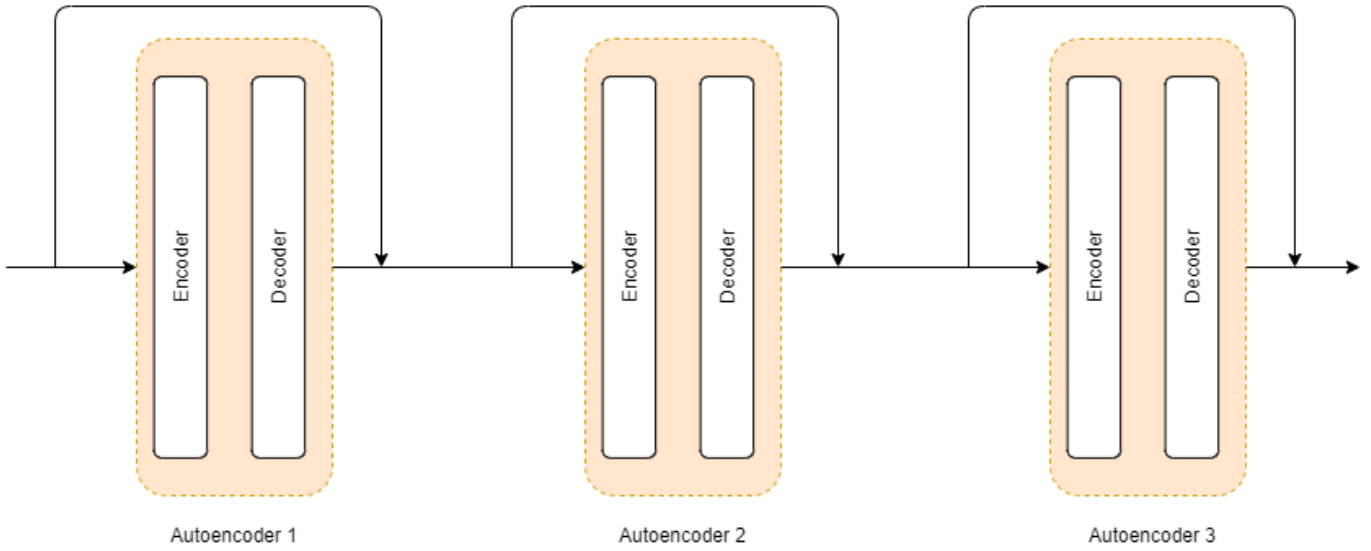
- Kod DAE-a, u procesu učenja, na ulaz se daju uzorci (vremenski niz, slika) koji sadrži različite razine dodanog (Gaussovog) šuma, a na izlazu originalni primjerak
- Mreža uči preslikavanje iz šumovitog u nešumoviti uzorak kroz unutarnje slojeve mreže i time efektivno uči filtrirati šum
- U praksi, pokazuje se da:
  - Kôd je često veće dimenzije od ostalih slojeva
  - Mreža ponekad nije simetrična – dekoder može imati više slojeva od kodera jer je lakše kodirati šumoviti uzorak nego dekodirati originalni uzorak
  - Za slike bolje je koristiti konvolucijske slojeve autoenkodera nego potpuno povezane (konvolucijski autoenkoder za uklanjanje šuma – *convolutional denoising autoencoder*)



https://towardsdatascience.com/denoising-autoencoders-dae-how-to-use-neural-networks-to-clean-up-your-data-cd9c19bc6915
https://keras.io/examples/vision/autoencoder/

Autoenkoderi – naslagani autoenkoderi

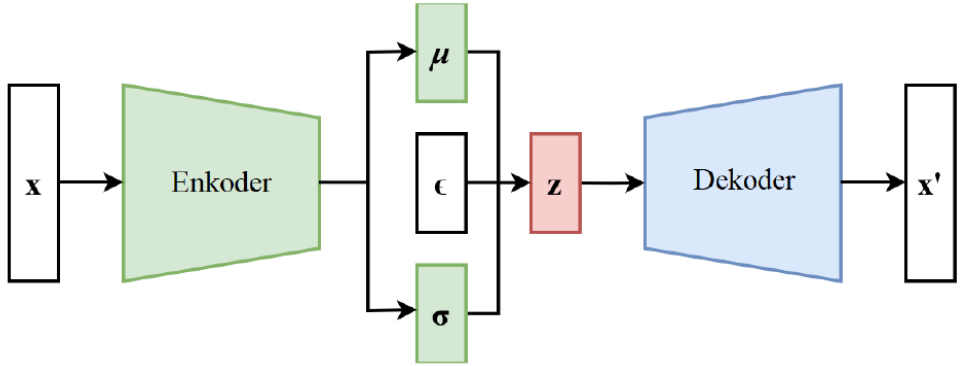
- **Naslagani autoenkoderi** (engl. *stacked autoencoder*)
- Koriste se za skupove podataka koji imaju kompleksne odnose među značajkama te stoga jedan autoenkoder ne može učinkovito smanjiti dimenzionalnost skupa ulaznih značajki
- Osim ulaznih podataka, svaki autoenkoder u nizu dobiva i predikciju od autoenkodera prošlog stupnja kako bi bolje naučio značajke
  - Time svaki idući autoenkoder može imati više ulaznih podataka od onog prethodnog što omogućuje učenje hijerarhije značajki
- Koristi se za nenadziranu ekstrakciju značajki, detekciju anomalija (učenje na normalnim podacima i sl.)



https://towardsdatascience.com/stacked-autoencoders-f0a4391ae282

Autoenkoderi – varijacijski autoenkoderi

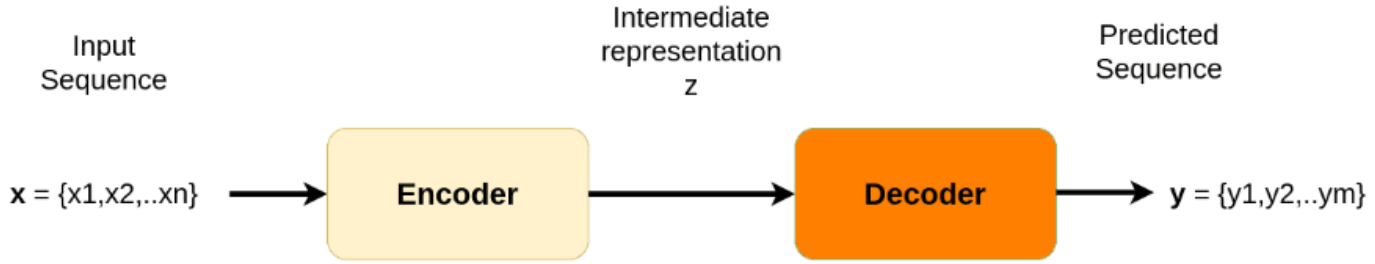
- **Varijacijski autoenkoderi** (engl. *variational autoencoder*), 2013.
- Koder preslikava ulaze u **lokacije u latentnom prostoru**, koji je opisan sa srednjom vrijednosti μ i stdev σ normalne razdiobe koja se uči iz podataka, ne postoji sloj koda
- Faktor šuma ε koristi se za **generiranje primjeraka** vrijednosti vektora *z* koji služi za rekonstrukciju
- Funkcija gubitka se regularizira **KL-divergencijom** između latentnog vektora generiranog iz naučene Gaussove latentne razdiobe i apriorne latentne razdiobe (normalne), a u samoj funkciji gubitka za **rekontrukciju** se koristi neka prikladna mjera, ovisno o primjeni (slika, tekst, num. podaci)
- Veličina latentnog vektora *n* je hiperparametar
- Naučeni varijacijski autoenkoder -> generativni model



$$KL\big(N(\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x), N(0,1)\big) = -\frac{1}{2}\sum_{k=1}^{n}(1 + \log\sigma_k^2 - \mu_k^2 - \sigma_k^2)$$

Transformeri

- Transformerska arhitektura primjenjuje se najčešće nad slijednim podacima, ali je najpoznatija primjena u **velikim jezičnim modelima** (engl. *large language models*) **za generiranje teksta**
- Transformerska arhitektura je **kodersko-dekoderska arhitektura** s određenim brojem (stogom) kodera i dekodera, koja djeluje prema mehanizmu pažnje (engl. *attention*), uklanjajući potrebu za rekurentnim blokovima
  - Mehanizam pažnje koristi poseban **sloj samopažnje** (engl. *self-attention layer*) kako bi uz fokus na određeni dio sekvence uzeo u obzir i njezin kontekst i time povezao međusobno dijelove sekvenci da bi se izračunala učinkovita reprezentacija pojma i njegovog konteksta
  - Dijelovi ulaznih podataka dobivaju **različite težine** u modelu u odnosu na trenutačno analizirani dio sekvence, ovisno o tome koliko su važni



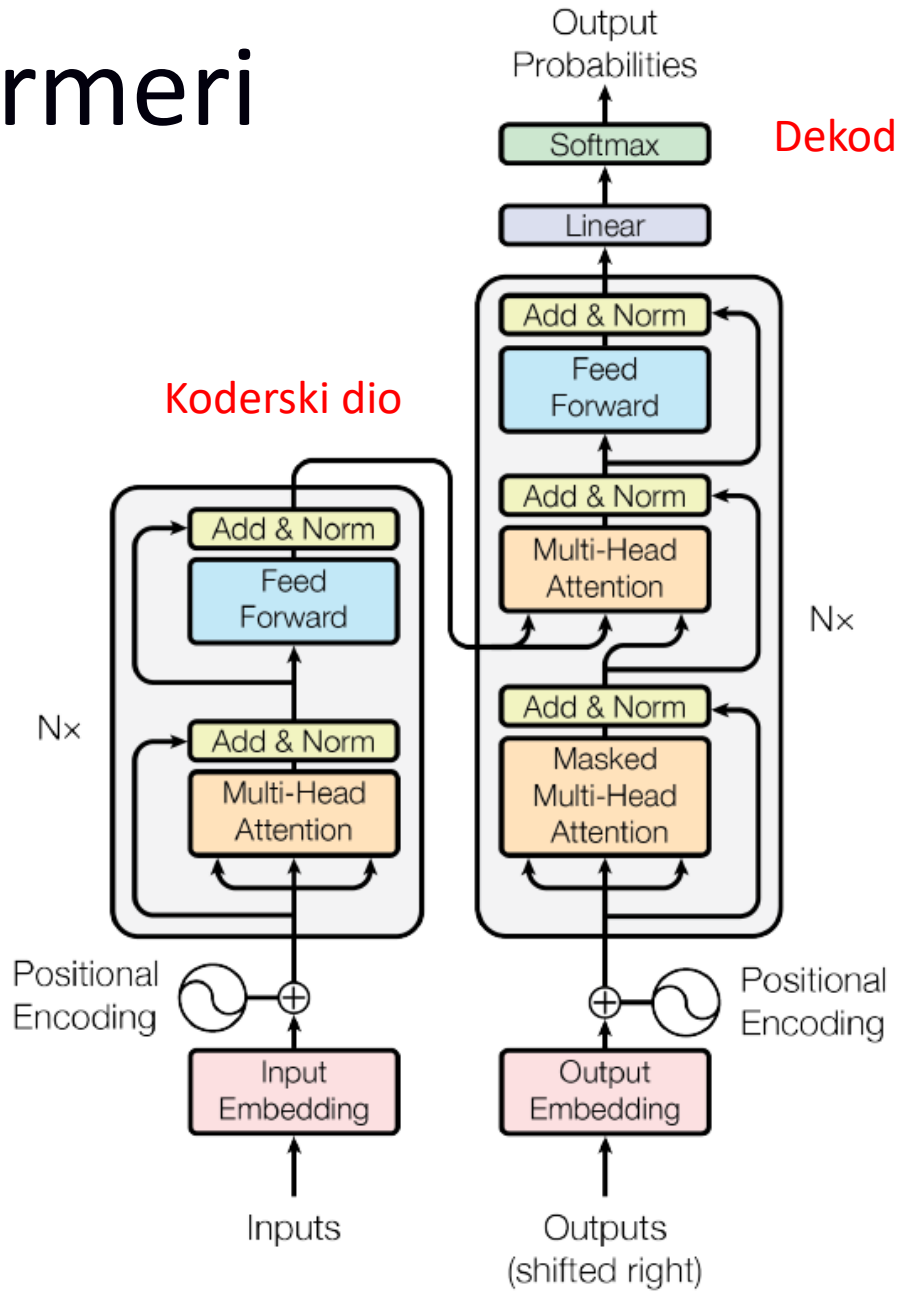
Najjednostavnija kodersko-dekoderska arhitektura za predviđanje nove sekvence (prije pojave transformera)

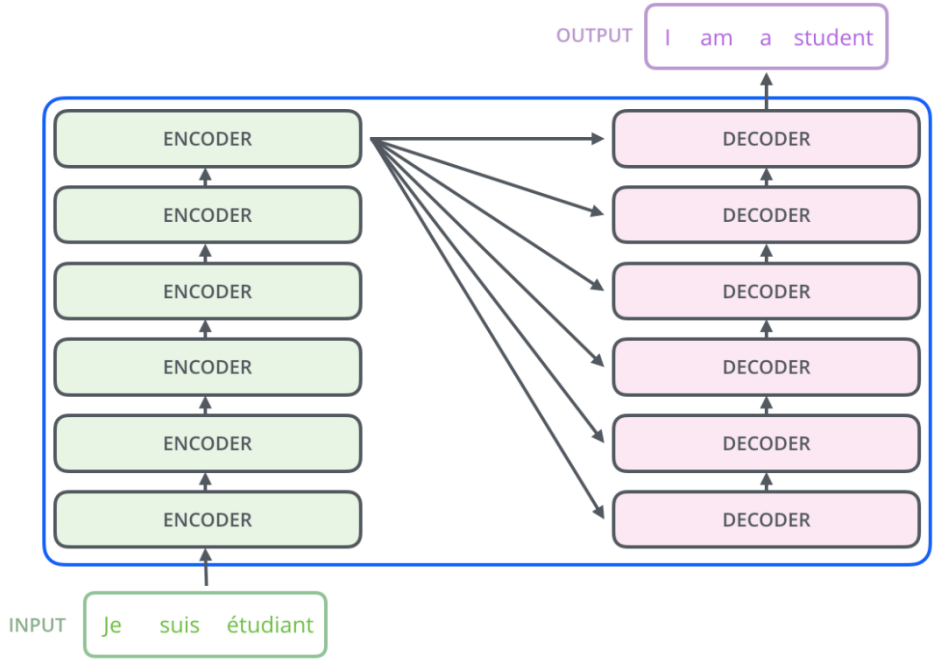
Vaswani et al. (2017), Attention is All you Need, Advances in Neural Information Processing Systems, vol. 30
http://nlp.seas.harvard.edu/2018/04/03/attention.html
https://theaisummer.com/attention/

Transformeri

Tianyang Lin, Yuxin Wang, Xiangyang Liu, Xipeng Qiu, A survey of transformers, AI Open, Volume 3, 2022, Pages 111-132, ISSN 2666-6510, https://doi.org/10.1016/j.aiopen.2022.10.001

Izvorna arhitektura transformera





Zadatak prevođenja s francuskog na engleski nakon što je model izgrađen

Transformeri – kratki opis izvorne arhitekture

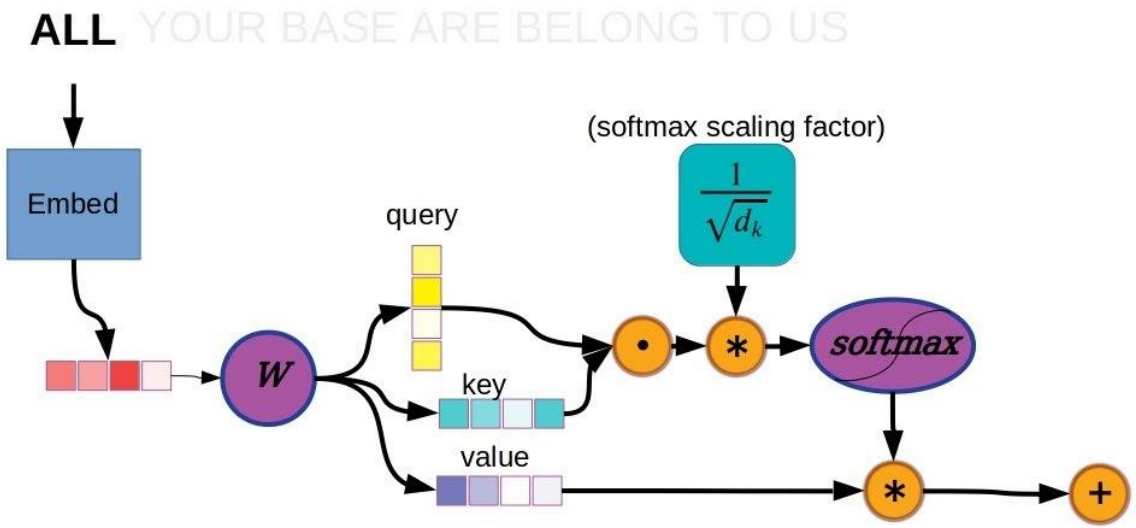
- Koderski dio sastoji se od stoga od ***N* identičnih slojeva** (koderskih slojeva), a svaki sloj od dva podsloja: sloja za ostvarenje samopažnje i višeslojnog perceptrona, dok su ulazi i izlazi iz svih podslojeva vektori iste dimenzije *M* (*M* je po defaultu 512)
- Zadana ulazna sekvenca (npr. rečenica koju se treba prevesti) dovodi se samo na ulaz prvog koderskog sloja te se iz nje određuju **tokeni** (npr. riječi, dijelovi riječi) za koje se potom određuje **vektorska reprezentacije** (engl. *embedding*) (svaki token se pretvori u vektor numeričkih vrijednosti dimenzije *M*)
- Svaki sloj prima listu tokena kao ulaz, koje obrađuje prvo kroz podsloj pažnje, a potom kroz podsloj višeslojnog perceptrona i potom šalje rezultate sljedećem sloju, proces se ponavlja sve dok se ne dosegne završni koderski sloj
- Izlaz od završnog koderskog sloja se potom predaje prvom dekoderskom sloju, a dalje svaki od dekodera u dekoderskom dijelu od *N* identičnih slojeva koristi izlaz od prošlog dekodera kao i informaciju od zadnjeg koderskog sloja za svoje učenje
- Nakon posljednjeg dekodera slijedi potpuno povezani sloj s aktivacijskom funkcijom *softmax* koja na izlazu daje distribuciju vjerojatnosti tokena po klasama, za koju se računa funkcija gubitka (i uspoređuje sa stvarnim tokenom u procesu učenja)

Transformeri – kratki opis izvorne arhitekture

- Broj kodera i dekodera varira od transformera do transformera, obično od 6 na više, a njihov broj utječe na kvalitetu učenja
- Arhitektura transformerske mreže uči se na sličan način kao i za većinu ostalih vrsta mreža: koristeći prolaz unaprijed i stohastički gradijentni spust s povratnom propagacijom pogreške
- Nastoji se naučiti dobro preslikavanje ulazne sekvence na izlaznu sekvencu podešavajući težine u unutarnjim slojevima koderskog i dekoderskog dijela
- Slojevi kodera i dekodera lančano propagiraju gradijent od zadnjeg dekoderskog sloja do prvog koderskog sloja, a parametri pojedinih slojeva se potom izračunavaju neovisno jedan o drugom, s mogućnošću paralelizacije
- Parametri koderskih slojeva su: težine povezane mreže, težine matrica za mehanizam samopažnje, težine normalizacijskog sloja i težine ulaznih vektorskih reprezentacija, dok dekoderski slojevi osim navedenih parametara dodatno imaju i parametre kodersko-dekoderske reprezentacije za mehanizam samopažnje

Transformeri – mehanizam pažnje

- Ulazna riječ se prvo tokenizira koristeći neku funkciju ugradnje (engl. *embedding function*) gdje se npr. riječ “ALL” zamijeni s numeričkim vektorom koji će biti ulaz u sloj pažnje (ta funkcija može se i naučiti iz podataka)
- Sloj pažnje (***W*** u dijagramu) računa tri vektora iz tog jednog tokena: *query, key* i *value* na temelju produkta vektora s odgovarajućim trima matricama ($\boldsymbol{W}^q$, $\boldsymbol{W}^k$, $\mathbf{W}^v$) koje se uče tijekom procesa učenja mreže
- Skalarni produkt *query* i *key* je skalar koji predstavlja relativnu težinu (skor) tokena na danoj poziciji
- Skor se razmatra i za ostale tokene u listi u odnosu na trenutno razmatrani token *i*, u tom se slučaju računa za svaki drugi token *j* skalarni produkt: query(i)*key(j)

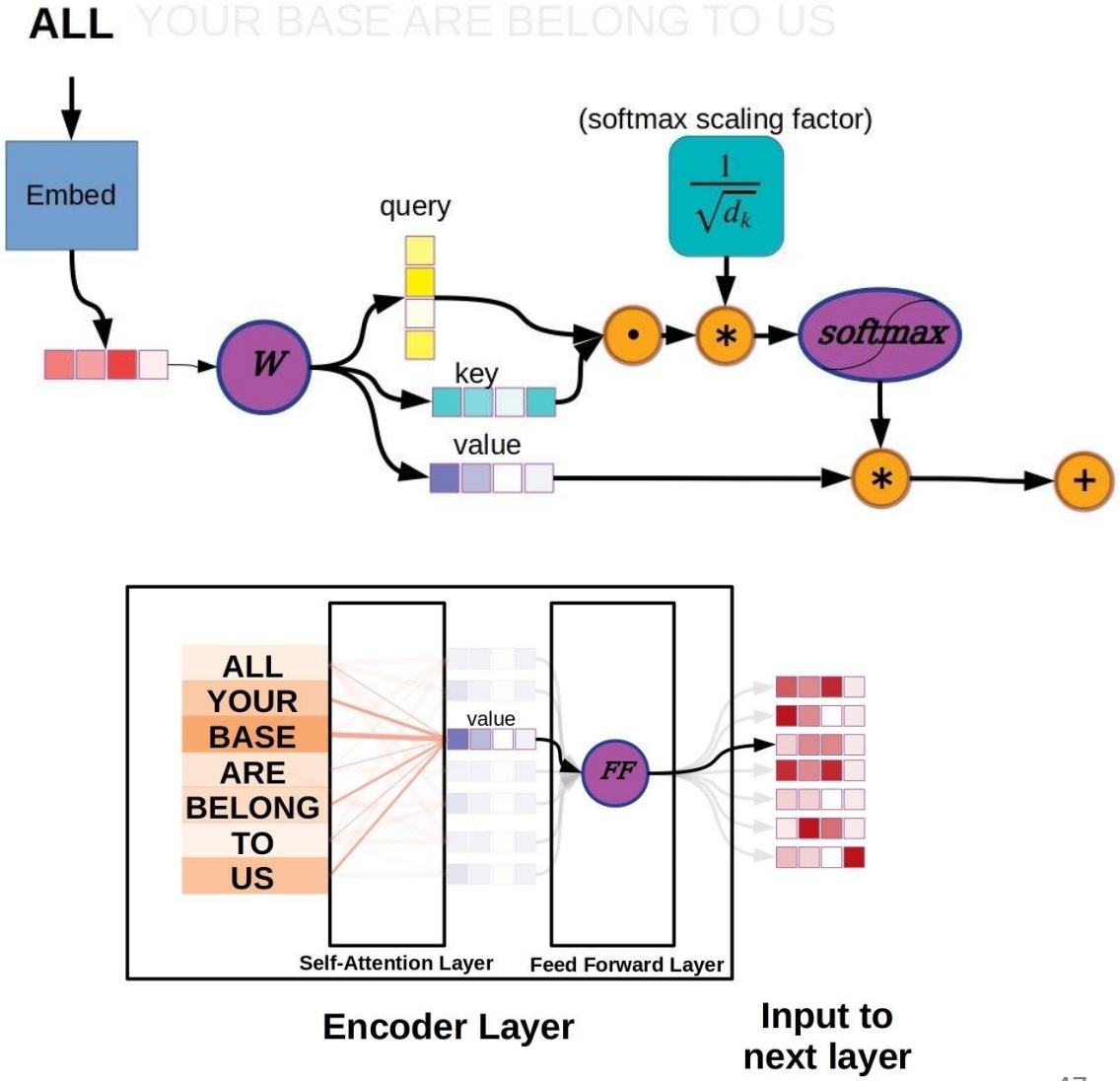


https://www.exxactcorp.com/blog/Deep-Learning/a-deep-dive-into-the-transformer-architecture-the-development-of-transformer-models

https://jalammar.github.io/illustrated-transformer/

Transformeri – mehanizam pažnje

- Mehanizam pažnje primijenjuje se jednako na svaki vektor u listi, tako da svaki drugi vektor ima isto svoj skalarni skor pažnje i skorove pažnje za njegov kontekst
- Skor pažnje se najprije skalira s faktorom *softmax scaling* (iznosi 1/8 ako je $d_k = 64$) i preda funkciji skaliranja *softmax* kako bi se sve težine sumirale do 1.0 i onda se množe s odgovarajućim vektorom *value*
- Vrijednosti svih vektorskih reprezentacija u listi, utežani prema skoru pažnje se zajedno sumiraju
- Rezultirajući vektor je nova reprezentacija pažnje za određeni token, koja se prosljeđuje potpuno povezanom sloju



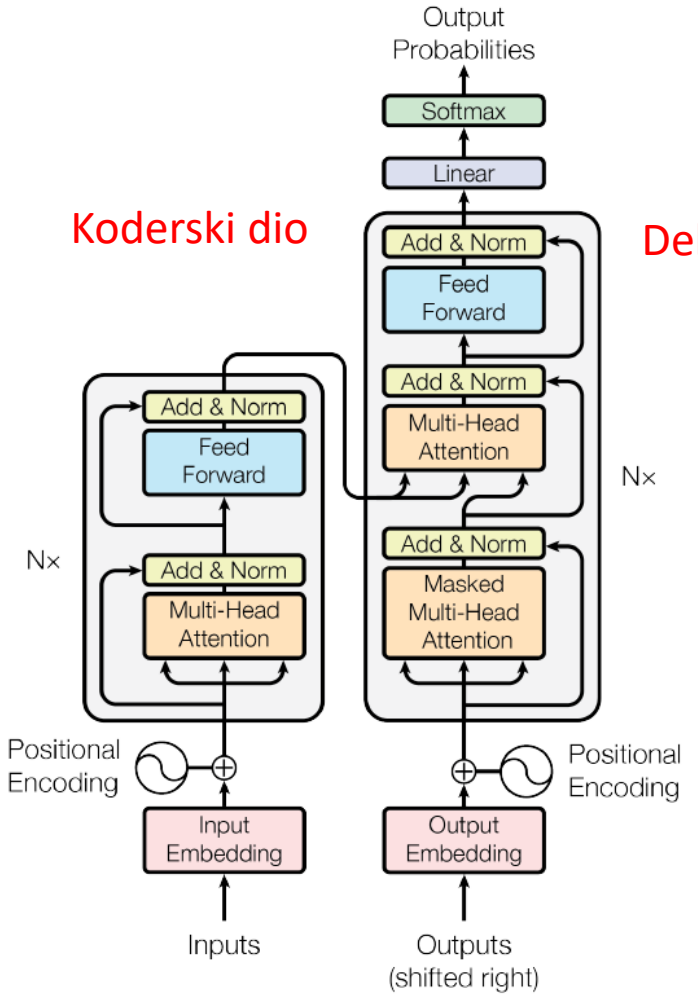
Transformeri

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,...,\text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$$\text{Attention}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

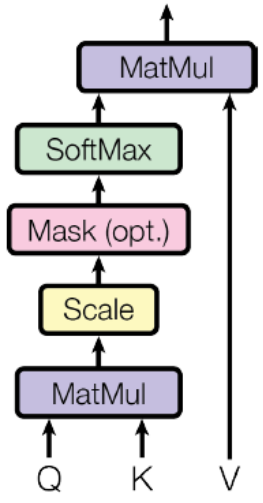
Koderski dio

Dekoderski dio

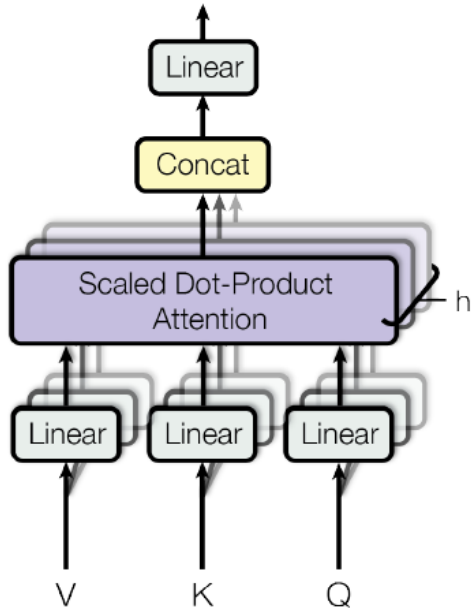


Transformerska arhitektura

Scaled Dot-Product Attention



Multi-Head Attention



- Mehanizam pažnje s više glava koristi po defaultu *h* = 8 paralelnih slojeva (glava) za čitanje ulaznog vektora dimenzije 512 (svaka glava čita 512/8 = 64 bita)
- Maskiranje mehanizma pažnje s više glava u dekoderskom dijelu koristi se za predviđanje vrijednosti na nekoj poziciji *i*, kako bi ono ovisilo samo o poznatim prethodnim vrijednostima, a ne o budućima (postavljajući buduće pozicije tokena na –inf.)

Načini korištenja temeljne transformerske arhitekture

- **Cjelokupna kodersko – dekoderska arhitektura (*default*)**
  - Koristi se za modeliranje “od sekvence do sekvence” (engl. *sequence-to-sequence*), npr. za prevođenje teksta
- **Samo koderska arhitektura**
  - Izlazi kodera koriste se za krajnju reprezentaciju ulazne sekvence
  - Koristi se za zadatke boljeg razumijevanja prirodnog jezika, npr. klasifikaciju teksta i druge vrste labeliranja sekvenci
- **Samo dekoderska arhitektura**
  - Uči se samo dekoder
  - Koristi se za različite vrste generiranja sekvenci, npr. jezično modeliranje, generiranje teksta, sumarizaciju
  - Autoregresijski uči novi izlaz na temelju prethodnih izlaza

Učenje transferom i fino podešavanje

- Tehnike koje su korisne za razne arhitekture mreža (konvolucijske, transformere) i široko su primjenjive
- **Učenje transferom** (engl. *transfer learning*)
  - Koristi **prednaučeni** (engl. *pre-trained*) model na **sličnom problemu** na novom problemu, npr. detekcija bicikala ako je prije model radio detekciju motora
  - Izlaz prednaučenog modela predstavlja ulaz u **novi klasifikacijski sloj** koji se uči na novom, dodatnom skupu podataka
  - Mogu se mijenjati postojeći slojevi ili dodavati novi slojevi, ali se težine jednog temeljnog dijela mreže fiksiraju kako se ne bi čitava arhitektura morala od početka učiti
- **Fino podešavanje** (engl. *fine-tuning*)
  - Slično, koristi se prednaučeni model, ali ga se želi **prilagoditi za određeni specifičan slučaj**, npr. klasifikacija voćaka ako je prednaučeni model klasificirao biljke
  - Za učenje se koristi dodatni skup podataka, ponekad usporedive, ali najčešće manje veličine od početnog
  - Uči se ponovno cijela mreža, ali se obično smanjuje stopa učenja na novom skupu da ne bude velikih promjena modela

Zaključak

- Duboki modeli imaju iznimno brzi razvoj u zadnjih desetak godina, često značajni znanstveni članci nisu prošli formalnu recenziju, ali se naširoko citiraju (što nije dobro!)
- Konvolucijske neuronske mreže uvijek treba prve razmotriti za analizu slike
- Rekurentne neuronske mreže i transformere treba razmotriti za analizu sekvencijalnih podataka (vremenski nizovi, tekst)
- Autoenkoderi imaju razne uspješne primjene i vrlo su računski učinkoviti zbog relativno jednostavne arhitekture
- Duboko učenje u raznim područjima postupno zamjenjuje tradicionalno strojno učenje, ali ne svugdje, pogotovo ne tamo gdje ima malo podataka ili su već dostupne značajke za učenje