Prepoznavanje višealfabetnog teksta na slici korištenjem dubokog učenja

Sažetak

U ovom radu je definirana arhitektura i provedena implementacija sustava za prepoznavanje višealfabetnog teksta na slici. Konkretno, opisani sustav je dizajniran za prepoznavanje latinice, arapskog i korejskog alfabeta. Sustav je implementiran u programskom jeziku Python, a za model je korišten vizualni transformer. Sa javnih internetskih stranica je prikupljen i pripremljen podatkovni skup za testiranje, a podatkovni skup za treniranje je umjetno generiran zbog nepostojanja dovoljno velikog javno dostupnog skupa. Model je, zbog ograničenih resursa, uspio samo u određenoj mjeri riješiti ovaj izazovni problem sa točnošću od 45.027%, ali glavni mu je doprinos to što predstavlja početnu točku za višealfabetno prepoznavanje teksta. Implementirani model ima mjesta za napredak, a najvažnija nadogradnja bi bila koristiti veći podatkovni skup za treniranje te isprobati veći skup metoda za pretprocesiranje slike. Trenirani su i modeli koji prepoznavaju zasebno svaki od tri navedena alfabeta. Modeli sa jednim alfabetom postižu bolje rezultate na tom individualnom alfabetu, ali se sami po sebi ne mogu koristiti kao sustav višealfabetnog prepoznavanja teksta jer im je potreban još jedan model koji će prvo odrediti o kojem se jeziku radi pa zatim proslijediti ulaznu sliku na odgovarajući model.

Ključne riječi: računalni vid, višealfabetno prepoznavanje teksta, duboko učenje, vizualni transformer, Python

Multi-alphabet Text Recognition Using Deep Learning

Abstract

In this project, the architecture and the implementation of the system for recognizing multi-alphabet text on the image is carried out. In particular, the described system is designed to recognize Latin, Arabic and Korean alphabets. The system was implemented in the Python programming language, and a visual transformer was used for the model. A dataset for testing was collected and prepared from public websites, and a dataset for training was artificially generated due to the lack of a large enough publicly available datasets. The model, due to limited resources, only partially managed to solve this challenging problem with an accuracy of 45.027%, but its main contribution is that it represents a starting point for multi-alphabet text recognition. The implemented model has room for improvement, and the most important upgrades would be to use a larger training dataset and to try a larger set of image preprocessing methods. Models that recognize each of the three listed alphabets separately were also trained. Single-alphabet models perform better on that individual alphabet, but they by themselves cannot be used as a multi-alphabet text recognition system because they need another model to first determine what language is found on an image and then pass the input image to the appropriate model.

Keywords: computer vision, Multi-alphabet text recognition, deep learning, visual transformer, Python

SADRŽAJ

1. Uvod **1**

2. Teorijska podloga i korištene tehnologije **3**

2.1. Računalni vid 3

2.2. Optičko prepoznavanje znakova 3

2.2.1. Pretprocesiranje slike 4

2.2.2. Lokalizacija teksta na slici 4

2.2.3. Standardni duboki model za optičko prepoznavanje znakova . 5

2.2.4. Duboki model za optičko prepoznavanje znakova temljen na transformer neuronskim mrežama 14

2.2.5. Arhitektura sustava za optičko prepoznavanje znakova 23

2.3. Izazovi višealfabetnog prepoznavanja teksta 23

2.4. Korištene tehnologije 24

3. Priprema skupova podataka **26**

3.1. Priprema podatkovnih skupova za latinicu 27

3.2. Priprema podatkovnih skupova za arapski alfabet 27

3.3. Priprema podatkovnog skupa za korejski alfabet 29

4. Programsko ostvarenje sustava za optičko prepoznavanje znakova i rezultati **31**

4.1. Detalji arhitekture korištenog vizualnog transformera 31

4.2. Implementacijski detalji vezani za treniranje modela 33

4.3. Rezultati 34

5. Zaključak **36**

Literatura **37**

1. Uvod

Današnji svijet je prepun tekstualnih informacija, a ispravna interpretacija istih je nužna za iskorištavanje pogodnosti koje nudi ljudska zajednica. U posljednjih nekoliko desetljeća su tekstualne informacije omogućile i veću povezanost geografski udaljenih područja koja tako postaju sve bliža. Njih svakodnevno susrećemo, a one se nerijetko nalaze napisane na različitim alfabetima, od kojih su jedna od najčešćih latinica, ćirilica, kineski, arapski te indijski alfabet [25]. Koegzistencija alfabeta je sveprisutna te nju treba njegovati kako bi se znanje širilo i preko jezičnih granica.

Napretkom digitalnog svijeta se broj dostupnih tekstualnih informacija povećava pa je sve izraženija potreba za robusnim sustavom koji može prepoznavati višealfabetni tekst. Sustavi za prepoznavanje teksta jednog alfabeta na slici su danas već dosta razvijeni i postižu točnost od 99% za strojno napisani tekst [2]. Prepoznavanje višealfabetnog teksta predstavlja veći izazov zbog varijacija u stilu, obliku i veličini pisanja znakova različitih alfabeta, ali ispravno prepoznavanje takvog teksta može znatno unaprijediti analizu dokumenata, prepoznavanje i prijevod jezika, snalaženje na novom geografskom području te pronalaženje informacija.

Metode dubokog učenja su se pokazale kao moćan alat za rješavanje složenih zadataka koji uključuju prepoznavanje raznih oblika te su tako omogućile značajan napredak u prepoznavanju teksta jednog alfabeta. Prepoznavanje višealfabetnog teksta sa sobom nosi jedinstvene izazove i nije još u toj mjeri istraženo područje kao prepoznavanju teksta jednog alfabeta, ali su neuronske mreže zasigurno najbolji alat koji danas postoji za rješavanje ovog problema. Standardna rješenja danas podrazumijevaju korištenje kombinacije konvolucijskih i povratnih neuronskih mreža, a nešto moderija koriste transformer neuronske mreže.

Ovaj rad je motiviran željom da se izgradi i ispita sustav za prepoznavanje višealfabetnog teksta. Sustav je zamišljen za prepoznavanje tri različita alfabeta, latinice, arapskog i korejskog alfabeta. Također, uz izgradnju sustava je i svrha rada prikupiti javno dostupne podatke za svaki od navedenih alfabeta te ih obraditi i na prikladan način pripremiti za metode dubokog učenja.

Ostatak ovog rada je organiziran na sljedeći način: u drugom poglavlju se opisuje teorijska podloga i korištene tehnologije, u trećem poglavlju je opisan postupak prikupljanja i pripreme skupova podataka za svaki alfabet, četvrto poglavlje opisuje detalje programskog ostvarenja kao i postignute rezultate te u petom poglavlju se iznosi zaključak.

2. Teorijska podloga i korištene tehnologije

U ovom poglavlju se definira teorijska podloga potrebna za razumijevanje arhitekture i programskog ostvarenja sustava te se navode i opisuju korištene tehnologije. Budući da obrada slike metodama strojnog učenja pripada području zvanom računalni vid, u poglavlju 2.1. se ono predstavlja. U poglavlju 2.2. se formalno definira prepoznavanje teksta sa slike kao proces zvan optičko prepoznavanje znakova. Poglavlje 2.3. opisuje izazove koje sa sobom nosi višealfabetno prepoznavanje teksta, a u poglavlju 2.4 se navode tehnologije potrebne za programsko ostvarenje višealfabetnog sustava za prepoznavanje teksta.

2.1. Računalni vid

Računalni vid je područje umjetne inteligencije koje se uglavnom bavi prepoznavanjem dvodimenzionalnih i/ili trodimenzionalnih predmeta[8]. Metode računalnog vida najčešće podrazumijevaju prikupljanje, analiziranje i razumijevanje podataka u obliku slike, ali se mogu pojaviti i podaci koji imaju isti dvodimenzionalni oblik kao i slika, a da nemaju istu semantiku, poput analize signala kroz vrijeme i frekvencijski pojas. Cilj ovih metoda je automatizirati poslove koje obavlja ljudski vizualni sustav. Popularne primjene su: detekcija i prepoznavanje objekata i lica, navigacija samovozećih (eng. *Self-driving*) automobila i robota, prepoznavanje bolesti s medicinskih slika te nadzor i sigurnost.

2.2. Optičko prepoznavanje znakova

Optičko prepoznavanje znakova (eng. *Optical character recognition - OCR*) je proces pretvaranja teksta sa slike ili skeniranog dokumenta u oblik razumljiv računalima

i strojevima[29]. Ovim procesom se izlučuje tekst koji može biti strojno ili rukom napisan u oblik koji se može pretraživati i uređivati pomoću računala. Kako bi se tekst uspješno prepoznao, potrebno je provesti niz koraka koji će biti opisani u sljedećim potpoglavljima. U potpoglavlju 2.2.1. se definira proces pretprocesiranja slika u svrhu njihove obrade metodama strojnog učenja, u potpoglavlju 2.2.2. se opisuje lokalizacija teksta, u potpoglavlju 2.2.3. se opisuje arhitektura standardnog dubokog modela za optičko prepoznavanje znakova, u potpoglavlju 2.2.4. se opisuje arhitektura dubokog modela za optičko prepoznavanje znakova temljen na transformer neuronskim mrežama te u potpoglavlju 2.2.5. se opisuje cjelokupna arhitektura sustava za prepoznavanje znakova.

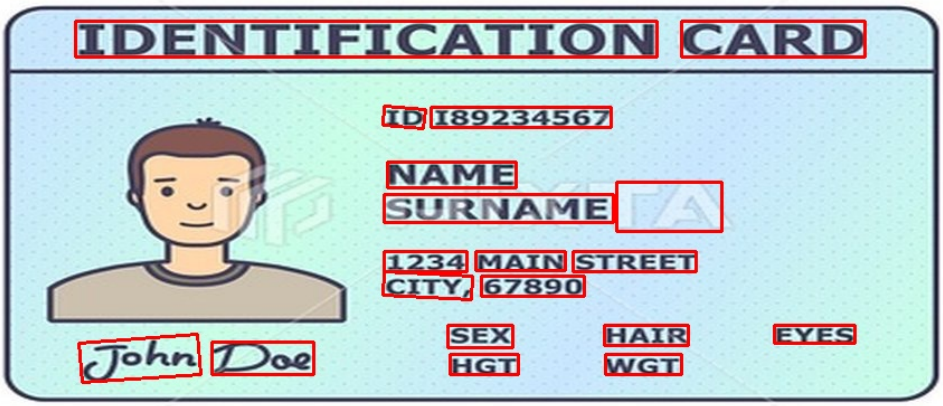
2.2.1. Pretprocesiranje slike

Metode strojnog učenja zahtijevaju da ulazni podatak bude u određenom obliku kako bi mogle nad istim provesti analizu ili predikciju pa tako i algoritmi obrade slike očekuju sliku u određenom obliku. Proces pripreme podataka za modele umjetne inteligencije nazivamo pretprocesianje podataka. Pretprocesiranje se provodi tako da se izabere skup metoda koje će se primijeniti na sliku kako bi ona bila spremna za prolazak kroz model. Neke od metoda koje se koriste za pretprocesiranje slika su: promjena veličine slike, povećanje ili redukcija šuma na slici, popravljanje kontrasta između piksela, normalizacija piksela, promjena broja kanala slike te rotacija slike za određeni kut. Promjena veličine slike se gotovo uvijek provodi zbog toga što modeli očekuju ulaznu sliku fiksne veličine. Promjena broja kanala slike se također mora provesti ako se npr. na ulazu očekuje crno-bijela slika pa se onda mora postaviti broj ulaznih kanala na 1. Ostale metode se pretežito koriste u fazi treniranja mreže. Često, ako nedostaje podataka nad kojima bi se mreža trenirala se provodi pretprocesiranje koje može umjetno stvoriti primjere za treniranje. Primjerice, od jedne slike se može stvoriti više slika tako da se originalnoj slici doda šum, promjeni kontrast ili se rotira. Stvaranje umjetnih podatak na ovaj način se naziva povećanje skupa podataka (eng. *Data augmentation*), a ono također služi i za treniranje robusnijeg modela koji može očekivati raznovrsnije slike na ulazu, tj. omogućuje povećanje generalizacijske moći modela.

2.2.2. Lokalizacija teksta na slici

Kako bi se iz slike ekstraktirao tekst, potrebno je na slici napraviti lokalizaciju teksta. Lokalizacija ili detekcija teksta je proces pronalaženja teksta ako je prisutan na slici, a nakon čega se tekst okružuje uskim pravokutnim graničnim okvirom[10]. Ovime se

što preciznije definiraju regije slika koje sadržavaju tekst sa svrhom eliminacije pozadinskih piksela kako oni ne bi negativno utjecali na prepoznavanje znakova. Moderni pristup lokalizacije teksta podrazumijeva korištenje konvolucijske neuronske mreže uz određeno pretprocesiranje slika prije nego što se pusti kroz mrežu. Slika 2.1 prikazuje primjer ispravne lokalizacije teksta na slici.



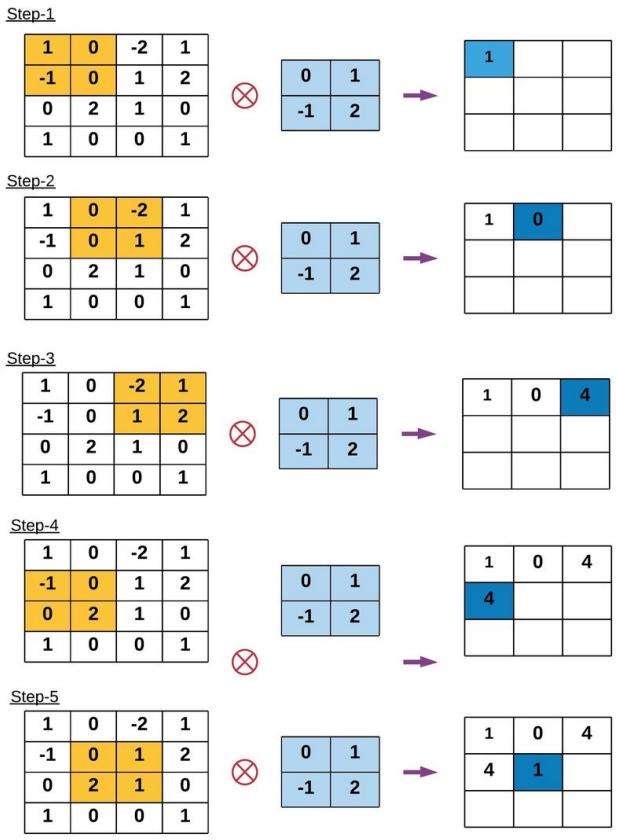
Slika 2.1: Primjer ispravne lokalizacije teksta na slici[24]

2.2.3. Standardni duboki model za optičko prepoznavanje znakova

Tradicionalni modeli za optičko prepoznavanje teksta prije ere dubokog učenja su se oslanjali na ručnu ekstrakciju značajki iz slika koja je često zahtijevala veliku količinu domenskog znanja pa i time činila model vrlo složenim. S druge strane, modeli koji koriste metode dubokog učenja su omogućili automatsko učenje značajki izravno iz podataka. Duboko učenje je omogućilo prepoznavanje složenih oblika, tj. vizualnih obrazaca te tako ostvarilo značajan napredak u točnosti i robusnosti u sustavima za prepoznavanje teksta. Pod standardnim modelom za optičko prepoznavanje znakova se podrazumijeva model dubokog učenja.

Standardna arhitektura za duboke modele optičkog prepoznavanja znakova se temelji na konvolucijskoj neuronskoj mreži (eng. *Convolutional Neural Network*) u kombinaciji s povratnom neuronskom mrežom (eng. *Recurrent Neural Network*). Ova je arhitektura poznata kao konvolucijska povratna neuronska mreža (eng. *Convolutional Recurrent Neural Network*) [13]. Ona se sastoji od konvolucijskih slojeva, povratnih slojeva te transkripcijskog sloja.

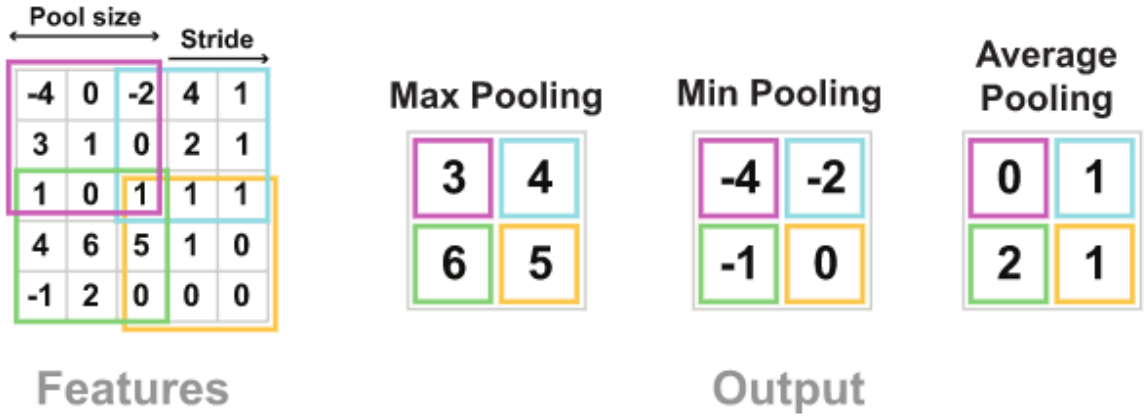
Konvolucijski slojevi su algoritam dubokog učenja koji je pogodan za obradu podataka rešetkaste strukture, poput slika ili vremenskih serija [13]. Često se primjenjuju pri rješavanju problema klasifikacije ili detekcije objekata na slici ili generiranja segmentacijske maske iz slike. Unutar ovih slojeva se provodi metoda konvolucije, zbog čega se i zovu tako. Konvolucija je matematička operacija koja provodi klizanje manjeg prozora preko rešetkastih podataka te pritom provode skalarno množenje po elementima između prozora i podataka. Prozor koji se kreće po podacima se još naziva i filtar ili jezgra. Ovom operacijom se provodi automatsko izvlačenje značajki iz slike. Jedan konvolucijski sloj obično provodi više različitih filtara nad ulaznom slikom, gdje svaki filtar ekstraktira druge značajke, tj. svaki filtar je zadužen za detekciju određenog uzorka sa slike. Na slici 2.2 se može vidjeti primjer provođenja operacije konvolucije nad slikom veličine četiri puta četiri piksela. Na slici se vidi prvih pet od sveukupno 9 koraka filtra koje je moguće napraviti nad ovom slikom.



Slika 2.2: Primjer provođenja operacije konvolucije na slici[16]

Između konvolucijskih slojeva se podrazumijeva korištenje i takozvanih *pooling* slojeva. *Pooling* sloj ima za cilj smanjiti prostorne dimenzije značajki iz konvolucijskih slojeva, ali uz to i zadržati najvažnije informacije u značajkama. slično kao i kod

konvolucije, *pooling* sloj koristi prozor koji se kreće po slici. Filtar je obično manje veličine, i to najčešće dva puta dva ili u nekim slučajevima tri puta tri piksela. Za razliku od konvolucijskog filtra, *polling* filtar neće provoditi skalarni umnožak, već će provoditi operacije poput pronalaska najveće vrijednosti (eng. *Max Pooling*), pronalaska najmanje vrijednosti (eng. *Min Pooling*) ili pronalaska srednje vrijednosti (eng. *Average Pooling*). S obzirom na to da ovaj sloj provodi redukciju prostornih dimenzija, on osigurava manju računalnu složenost modela te i manji broj parametara koji se trebaju učiti, a to sve zajedno u konačnici pomaže pri izbjegavanju prenaučenosti. Na slici 2.3 se može vidjeti primjer provođenja operacije *pooling* s filtrom veličine tri puta tri piksela na tri načina: pronalaskom najveće, najmanje i srednje vrijednosti.

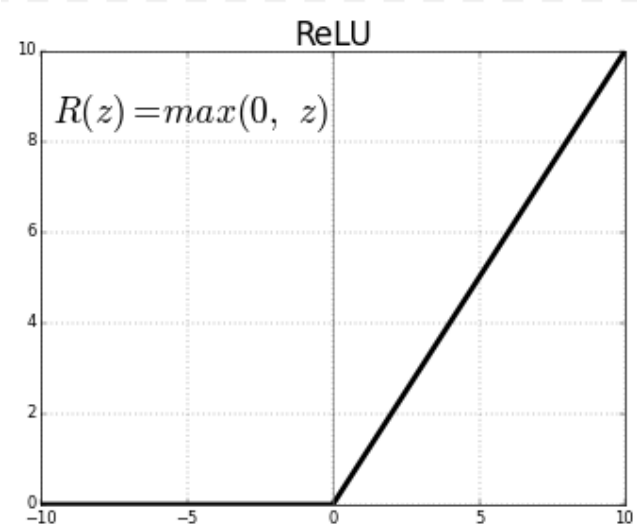


Slika 2.3: Primjer provođenja tri različite varijante operacije *pooling*[11]

Također, između konvolucijskih slojeva se primjenjuju aktivacijske funkcije. Obično se za aktivacijsku funkciju koristi nelinearni ReLu (eng. *Rectified Linear Unit*) koji omogućuje modelu da nauči složene odnose u podacima. Na slici 2.4 se može vidjeti izgled i formula za ReLu funkciju.

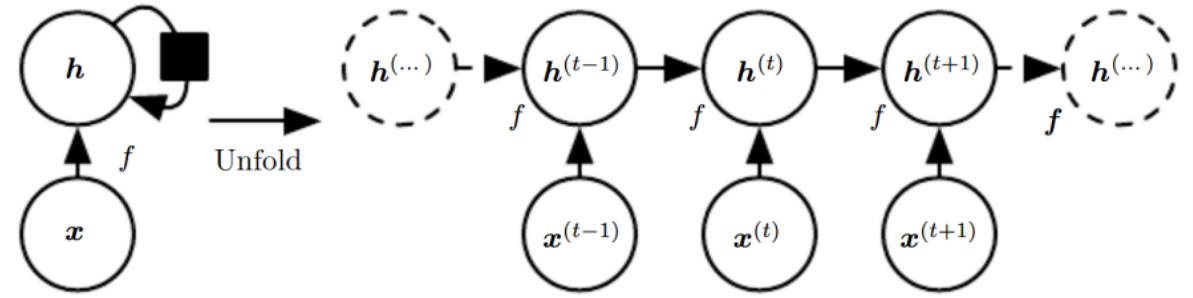
Nakon konvolucijskih slojeva se dobivaju značajke koje reprezentiraju ulaznu sliku te se one dalje mogu koristiti u drugim metodama strojnog učenja.

Povratni slojevi su osmišljeni s idejom obrade slijeda podataka proizvoljne duljine, gdje isto broj parametar mreže ne ovisi o duljini ulaza. Takvi slojevi su svjesni poretka ulaznih podataka. [26] Kako bi slojevi mogli obavljati navedeno, oni u sebi čuvaju takozvano skriveno stanje (eng. *hidden state*) koje ažuriraju kroz cijeli niz ulaznih podataka. Skriveno stanje se označava s oznakom $h^{(t)}$, gdje t u eksponentu označava da se radi o skrivenom stanju u trenutku t ili skrivenom stanju nakon t ulaznih podataka. Početno skriveno stanje se izvodi iz prvog podataka u ulaznom nizu i početnog skrivenog stanja koje se obično inicijalizira kao vektor nula, nad kojim se primjeni



Slika 2.4: ReLu funkcija[1]

određena funkcija. Dalje se skrivena stanja izvode iz trenutnog skrivenog stanja i novog podataka u nizu. Slika 2.5 prikazuje ideju skrivenih stanja u povratnim slojevima.

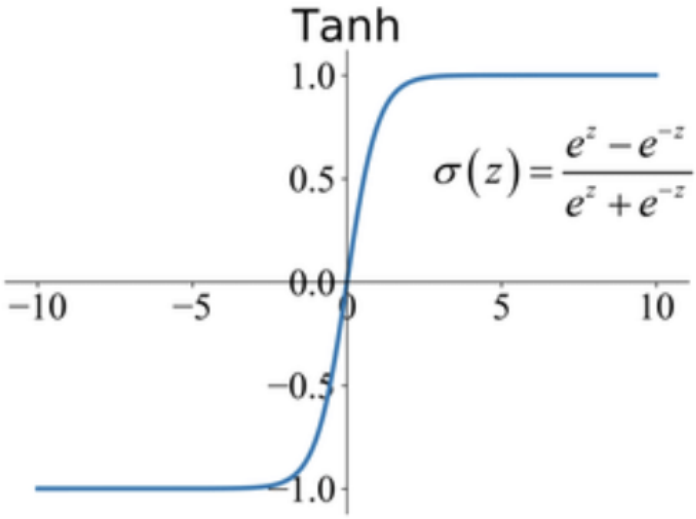


Slika 2.5: Skriveno stanje u povratnim slojevima i njegovo ažuriranje kroz niz podataka[26]

Skriveno stanje u trenutku t se računa po formuli 2.1.

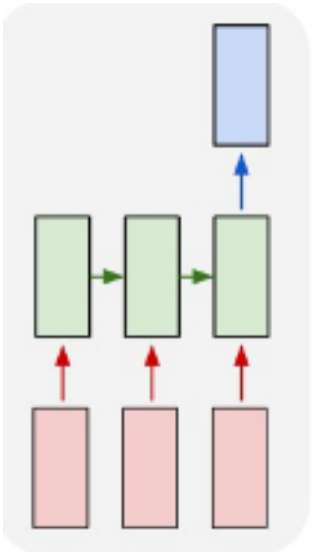
$$h^{(t)} = tanh(W_{hh}h^{(t-1)} + W_{xh}x^{(t)} + b_h)[9] \tag{2.1}$$

Oznaka W u formuli označava da se radi o matrici težina koje pretvaraju određeni element u prostor značajki. Prva matrica W_{hh} je matrica dimenzija $h * h$ i ona pretvara prethodno stanje u prostor značajki, a matrica W_{xh} je matrica dimenzija $x * h$ i ona pretvara trenutni ulazni element u prostor značajki. Element b_h predstavlja vektor pristranosti (eng. *bias vector*) i on je već u prostoru značajki. Nad zbroj elemenata u prostoru značajki se obično primjenjuje nelinearna funkcija tangens hiperbolni, čiju formulu i izgled prikazuje slika 2.6.



Slika 2.6: Tangens hiperbolni[6]

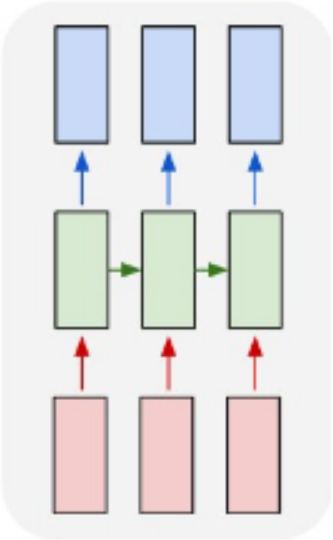
Ključna komponenta povratnih slojeva je ponovno izračunavanje skrivenog stanja kada se pojavi novi ulaz, a ono djeluje kao memorija za pamćenje informacija o prethodnim ulazima. Povratni slojevi se koriste za tri zadatka: klasifikacija slijeda (eng. *sequence classification*) , označavanje slijeda (eng. *sequence labeling*) te za slijed u slijed (eng. *sequence-to-sequence*) zadatak. Klasifikacija slijeda je zadatak određivanja klase koja se odnosi na cijeli ulazni niz podataka. Primjer ovog zadatka je predikcija sentimenta na temelju teksta, a primjer izgleda takve povratne mreže se vidi na slici 2.7.



Slika 2.7: Primjer klasifikacija slijeda[23]

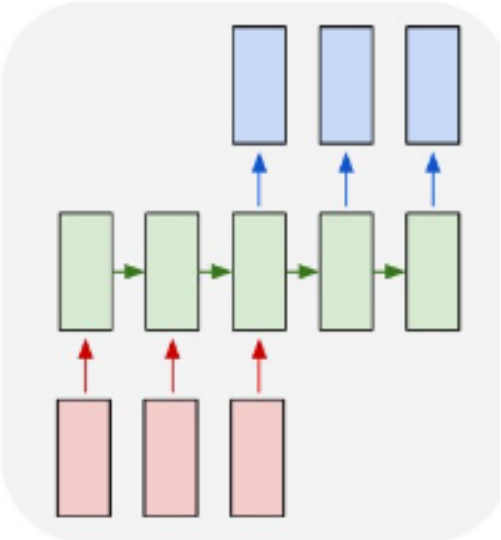
Označavanje slijeda je problem koji izbacuje predikciju za svaki podataka u ulaz-

nom nizu. Primjer ovog problema je određivanje vrsta riječi u tekstu i optičko prepoznavanje znakova, a na slici 2.8 se vidi primjer izgleda povratne mreže koja rješava takav zadatak.



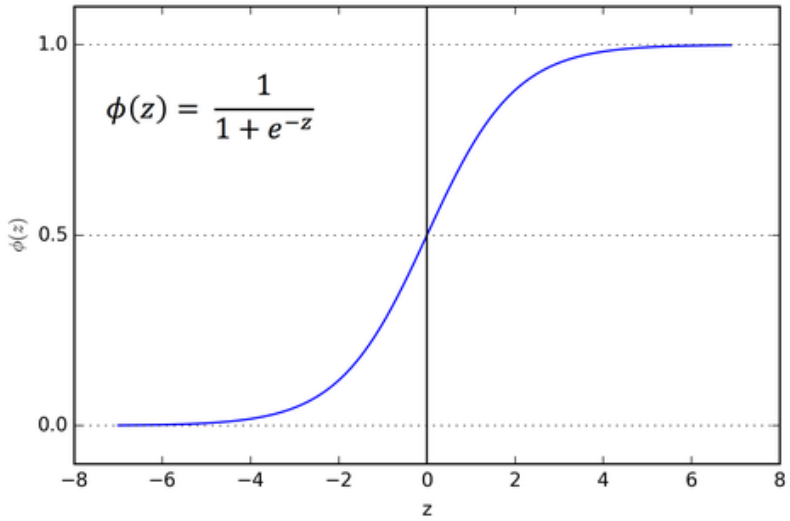
Slika 2.8: Primjer označavanje slijeda[23]

Slijed u slijed zadatak se bavi generiranjem slijeda izlaznih elmenata iz slijeda ulaznih elemenata, gdje izlazni slijed ne mora biti iste duljine. Primjer ovog zadatka je generiranje sažetka teksta, a primjer povratne mreže koja obavlja ovaj zadatak se vidi na slici 2.9.



Slika 2.9: Primjer za slijed u slijed zadatak[23]

Na ovaj način definirani povratni slojevi pate od problema zvanog nestajući gradijent (eng. *vanishing gradient*). Nestajući gradijent je problem koji se može pojaviti



Slika 2.10: Sigmoida[27]

tijekom faze treniranja dubokih neuronskih mreža, tj. mreža s puno slojeva, a okarakteriziran je smanjenjem vrijednosti gradijenta dok se propagira unazad i to do jako niskih vrijednosti blizu nule. Dakle, gradijent se krene propagirati od kraja mreže prema početku i na tom putu "nestane", a budući da su gradijenti zaslužni za učenje mreže, onda se početni slojevi mreže neće moći trenirati, a upravo oni su često zaslužni za detekciju osnovnih uzoraka u podacima. Problem nestajućeg gradijenta se pojavljuje prilikom korištenja određenih aktivacijskih funkcija koje ulaznu vrijednost zgnječe u uski raspon brojeva, kao što to radi sigmoida na raspon od 0 do 1 (slika 2.10) ili tangens hiperbolni na raspon od -1 do 1. [32]

Kako bi se riješio problem nestajućeg gradijenta, razvijene su varijante povratnih slojeva, kao što su ćelije s dugoročnom memorijom (eng. *Long Short Term Memory*). Razlika s obzirom na osnovnu definiciju povratnih slojeva je da ove ćelije koriste $h^{(t)}$ samo za izlaz u trenutnom koraku, a za pamćenje svih ulaza koriste vektor stanja ćelije $c^{(t)}$ koji se s ulaznim podacima ažurira [26]. Pomoću vektora stanja ćelije se želi rasteretiti posao skrivenog stanja i tako se omogućio bolji prijenos znanja kroz slojeve. Vektor stanja ćelije se ažurira po formuli 2.2.

$$c^{(t)} = f^{(t)} * c^{(t-1)} + i^{(t)} * \hat{c}^{(t)} \tag{2.2}$$

Formula se sastoji od zbroja dva umnoška, gdje prvi umnožak predstavlja udio informacije iz prošlosti koje prenosimo u novo stanje, a drugi umnožak označava udio nove informacije u stanju ćelije. Varijabla $f^{(t)}$ predstavlja udio informacije koju prenosimo iz prošlog stanja pa se ona množi s prošlim stanjem $c^{(t-1)}$, a računa se po formuli 2.3.

$$f^{(t)} = \sigma(W_{fhh}h^{(t-1)} + W_{fxh}x^{(t)} + b_{fh}) \tag{2.3}$$

Varijabla $f^{(t)}$ ima svoj skup težina koje će se trenirati te koristi sigmoidu kao aktivacijsku funkciju jer ona translatira vrijednosti u vjerojatnosni raspon od 0 do 1 koja će semantički predstavljati udio starih informacija koje se prenose dalje. Varijabla $i^{(t)}$ predstavlja udio novih informacija koje se ugrađuju u stanje ćelije, a računa se pp formuli 2.4.

$$i^{(t)} = \sigma(W_{ihh}h^{(t-1)} + W_{ixh}x^{(t)} + b_{ih}) \tag{2.4}$$

Ova varijabla, također, ima svoje težine za učenje i koristi sigmoidu za aktivacijsku funkciju. Još se varijabla $c^{(t)}$ računa po formuli 2.5.

$$\hat{c}^{(t)} = \tanh(W_{chh}h^{(t-1)} + W_{cxh}x^{(t)} + b_{ch}) \tag{2.5}$$

Ova varijabla predstavlja međurezultat ažuriranja stanja ćelije, tj. nove informacije koje ekstraktiramo iz ulaznog podatka te ona za to koristi tangens hiperbolni kao aktivacijsku funkciju. Novo stanje ćelije se tako ažurira po formuli 2.1.

Trenutni izlaz iz ćelije, tj. njezino skriveno stanje $h^{(t)}$ se računa po formuli 2.6.

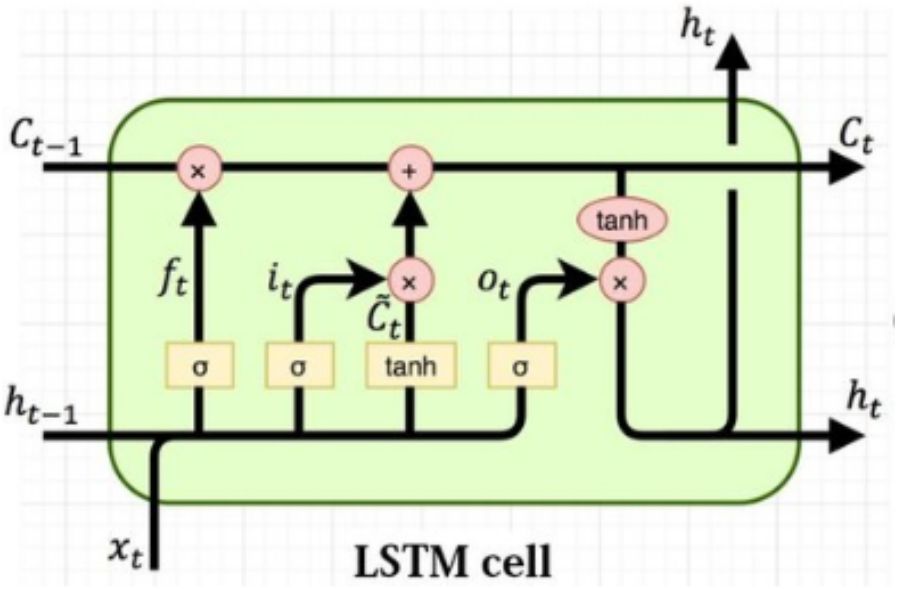
$$h^{(t)} = o^{(t)} * tanh(c^{(t)}) \tag{2.6}$$

$c^{(t)}$ je trenutno stanje ćelije dobiveno prijašnjim formulama, a $o^{(t)}$ predstavlja izlaznu propusnicu koja se računa po formuli 2.7.

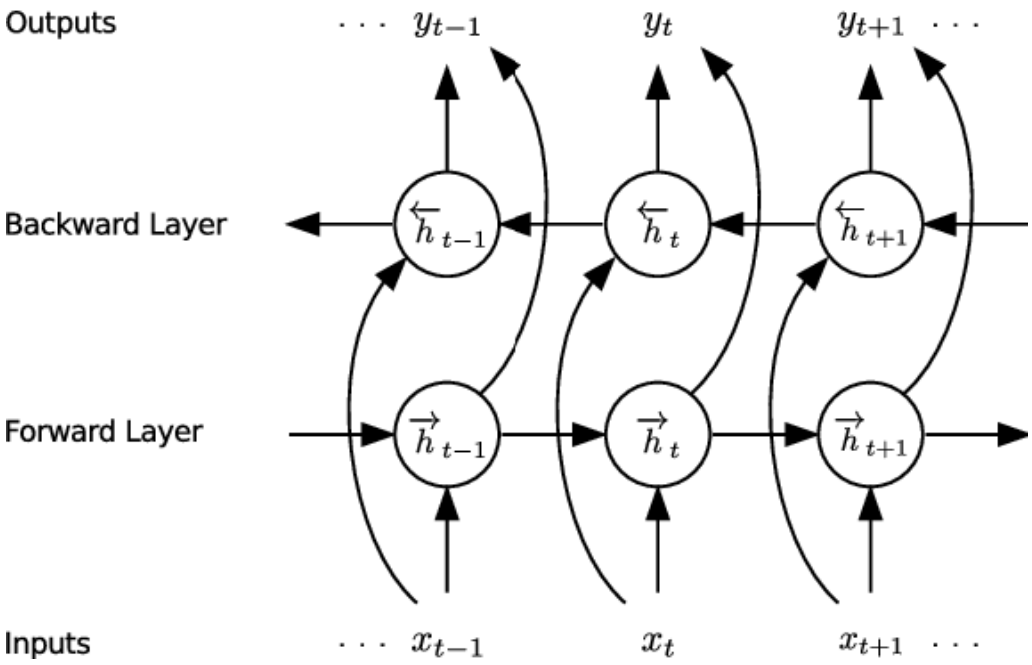
$$o^{(t)} = \sigma(W_{ohh}h^{(t-1)} + W_{oxh}x^{(t)} + b_{oh}) \tag{2.7}$$

Pomoću navedenih formula, ove ćelije uspijevaju pronaći složenije veze u podacima od osnovnih povratnih slojeva, ali zbog toga imaju četiri puta više parametara. Na slici 2.11 se može vidjeti opisani izgled ćelije s dugoročnom memorijom.

Dodatno, kako bi se unaprijedili povratni slojevi, su osmišljeni dvosmjerni povratni slojevi (eng. *Bidirectional Recurrent Neural Networks*) [7]. Ovi slojevi obrađuju podatke u naprijed i u nazad kako bi mogli iskoristiti informacije iz prošlih i budućih ulaza. Način na koji su se prvotno definirali povratni slojevi je uzimao u obzir samo informacije iz prethodnih ulaznih podataka, ali su ovi slojevi pokazali da i buduće informacije mogu znatno pridonijeti točnosti slojeva. Dvosmjerni povratni slojevi se sastoje od dva povratna sloja, gdje svaki obrađuje svoj smjer, koji održavaju vlastito



Slika 2.11: Arhitektura ćelije s dugoročnom memorijom[30]



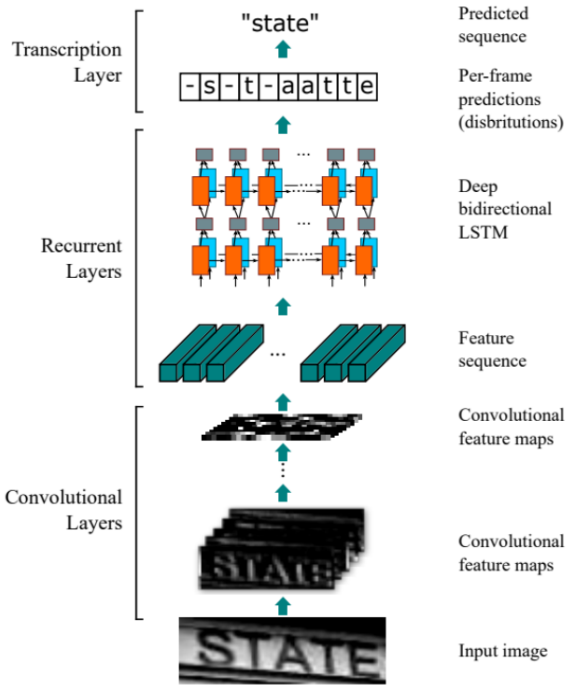
Slika 2.12: Primjer arhitekture dvosmjernih povratnih slojeva[18]

skriveno stanje. Oba sloja primaju podatke u isto vrijeme, a kako bi se koristili za generiranje izlaza, kombiniraju se pomoću konkatenacije ili vektorskog zbrajanja.

Transkripcijski sloj obrađuje značajke na ulazu te ih pretvara u slijed znakova[28]. Ovaj se sloj bavi problemom pronalaska najvjerojatnijeg znaka na temelju ulaznih značajki. Obično se ovaj sloj sastoji od potpuno povezanog sloja s aktivacijskog funkcijom

softmax na izlazu, koja generira vjerojatnosnu distribuciju po znakovima za svaku poziciju u ulaznom nizu. Preko distribucije se odabiru najvjerojatniji znakovi.

Dakle, standardna arhitektura modela za optičko prepoznavanja znakova se sastoji od konvolucijskih slojeva, povratnih slojeva i transkripcijskog sloja na izlazu. Prvi dio modela sačinjavaju konvolucijski slojevi koji obavljaju ekstraktiranje značajki iz ulazne slike u obliku okvira. Zatim, povratni slojevi na ulaz primaju niz okvira značajki iz konvolucijskih slojeva te ih obrađuju tako da obavljaju predikciju za svaki okvir, čiji broj može biti varijabilan pa se i zbog toga koriste povratni slojevi. Na kraju, transkripcijski sloj pretvara predviđanja za svaki okvir u niz znakova. Ovo je standardna arhitektura (slika 2.13), ali se u praksi nalaze razne varijante na nju, npr. s različitim brojem slojeva, različitim aktivacijskim funkcijama ili novim tipovima slojeva.



Slika 2.13: Primjer standardnog dubokog modela za prepoznavanje teksta na slici[5]

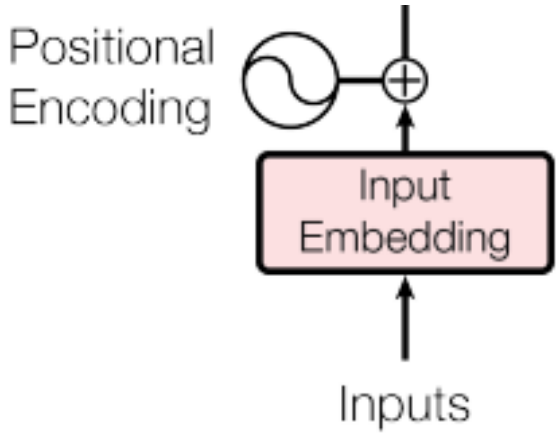
2.2.4. Duboki model za optičko prepoznavanje znakova temljen na transformer neuronskim mrežama

U ovom potpoglavlju se opisuje rad novijeg modela za optičko prepoznavanje znakova. Ovakav model se bavi nekim od problema s kojima se susreće standardni model, ali s time unosi dodatnu složenost koja rezultira sa sporijim procesom treniranja. U

nastavku će biti opisana transformer neuronska mreža (eng. *Transformer*) te njezina funkcija u rješavanju problema optičkog prepoznavanja znakova.

Transformer neuronska mreža

Transformer neuronske mreže su predstavljene u znanstvenom radu zvanom *Attention Is All You Need*. Služe za obradu sekvencijalnih podatka i hvatanje dalekih zavisnosti između podataka, a glavni koncept koji to omogućuje je mehanizam pozornosti (eng. *attention*) [31]. Pozornost stavlja veću važnost na određene elemente ulaznog niza u određenom kontekstu, tj. određuje koje su ulazne informacije bitne za izlaz u nekom trenutku. Pomoću ovog mehanizma, mreža istovremeno može vidjeti cijeli ulazni niz podataka, a to je razlika s obzirom na povratne slojeve koji obrađuju ulazni niz slijedno po podacima. Dakle, prednost transformer neuronski mreža je da mogu paralelno obrađivati podatke, a takva arhitektura bolje iskorištava prednosti modernih računala koja u sebi imaju hardverske akceleratore poput GPU-a (eng. *Graphics processing unit*) i TPU-a (eng. *Tensor Processing Unit*) jer oni upravo omogućuju paralelizaciju. Također, zbog toga što transformer neuronske mreže mogu istovremeno vidjeti sve ulaze, mogu i bolje uhvatiti zavisnosti u podacima nego povratni slojevi, pogotovo kada se radi o jako dalekim međuzavisnim podacima.



Slika 2.14: Priprema ulaznih podataka za transformer neuronsku mrežu[31]

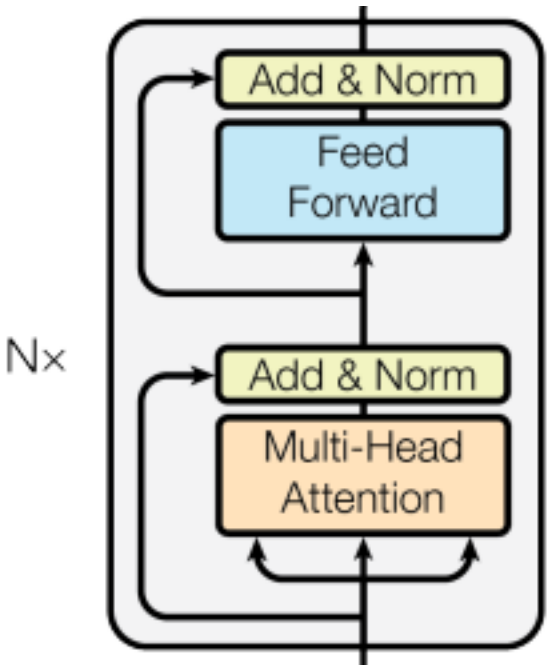
Ulazni sekvencijalni podaci se prvo trebaju pretvoriti u odgovarajući oblik kako bi se mogli obraditi transformer neuronskom mrežom, a taj je proces predstavljen slikom 2.14. Na slici se vidi da se iz ulaznih podataka prvo generiraju ugrađeni vektori (eng. *embeddings*), tj. ulazni se podaci pretvaraju u numeričke vektore određenog visoko dimenzionalnog prostora. U gore navedenom radu, kao što je i praksa u većini slučajeva, se koriste gotovi modeli za generiranje ovih vektora te je za njihove svrhe korišten 512

dimenzionalni prostor. Nakon toga se urađenim vektorima dodaje pozicijsko kodiranje (eng. *Positional Encoding*) koje omogućuje modelu da iskoristi informaciju o poziciji pojedinih ulaza. Informacija o poziciji se ubacuje pomoću vektora koji ima istu dimenziju kao ugrađeni vektor kako bi se mogao samo zbrojiti s njime. Standardni način pozicijskog kodiranja se provodi formulama 2.8.

$$\begin{aligned} PE_{(pos,2i)} &= \sin(\frac{pos}{10000^{2i/d}}) \\ PE_{(pos,2i+1)} &= \cos(\frac{pos}{10000^{2i/d}}) \end{aligned} \tag{2.8}$$

U formulama 2.8, PE označava izraz *Positional Encoding* i predstavlja vektor vrijednosti koji se nadodaje na ugrađeni vektor, *pos* predstavlja poziciju elementa u nizu, *i* predstavlja poziciju u višedimenzionalnom vektoru dimenzije *d*.

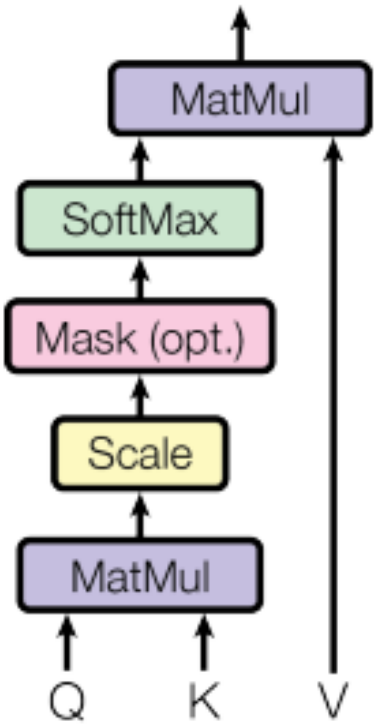
Nakon što je ulazni podatak pretvoren u numerički vektor i nakon što je u taj vektor ugrađena informacija o poziciji tog podatka u nizu, ugrađeni vektor ulazi u prvi od N koder slojeva (Slika 2.15).



Slika 2.15: Koder sloj transformer neuronske mreže[31]

Svaki sloj kodera se sastoji od dva podsloja, gdje prvi obavlja samopozornost s više glava (eng. *Multi-Head Self-Attention Mechanism*), a drugi je podsloj unaprijedna neuronska mreža. Na izlaz od oba sloja se rezidualnom vezom dodaje vektor koji ulazi u taj sloj kako bi sigurno bilo očuvano pozicijsko kodiranje, tj. da se ne bi poremetilo u nekom sloju, te se zatim suma tih vektora normalizira po sloju (eng. *Layer Normalization*). Sloj samopozornosti s više glava pretvara ugrađeni vektor u vektor pozornosti

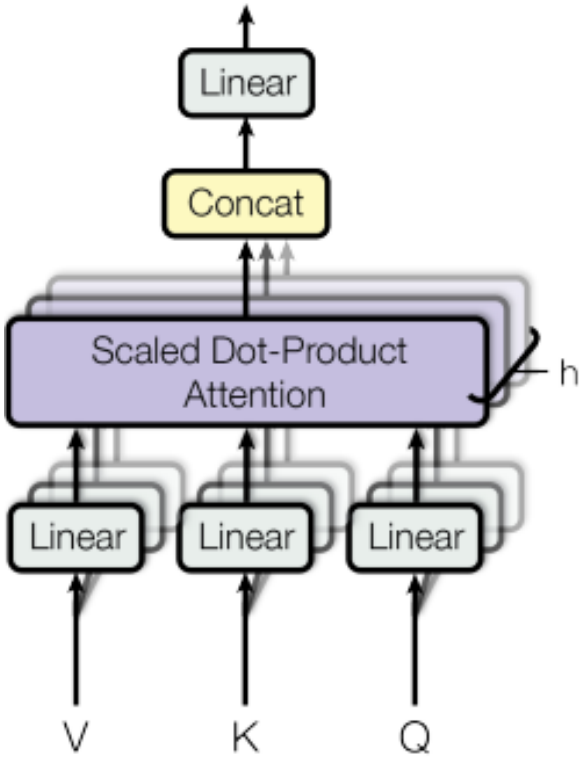
u kojem je zapisana informacija o tome u kojoj je mjeri pojedini ulaz povezan s drugim ulazom. Upravo zbog toga što se određuje povezanost samo između uzlaza, a ne između ulaza i izlaza, se ovo zove samopozornost (eng. *Self-Attention*). Mehanizam pozornosti određuje ove povezanosti koristeći tri vektora: vektor upita (eng. *query*), ključa (eng. *key*) i vrijednosti (eng. *value*). Svaki od ta tri vektora se dobije matričnim množenjem ugrađenog vektora s pripadajućom matricom težina i te se vrijednosti generiraju za svaki ulazni podatak. Proces dobivanja povezanosti je sljedeći: Vektori upita i ključeva za svaki ulazni podatak se kombiniraju u matrice koje pomnože kako bi se dobila kvadratna matrica dimenzije jednake broju ulaznih podataka. Vrijednosti u toj matrici predstavljaju mjeru povezanosti svake riječi sa svakom. Matrica se zatim skalira kako bi se osigurali stabilni gradijenti i time spriječile eksplozije gradijenata na jako velike vrijednosti. Matrica se još provede kroz softmax funkciju kako bi se mjere povezanosti postavile u raspon od 0 do 1. Na kraju se matrica pomnoži s vektorom vrijednosti za svaki ulazni podatak i time se dobiva informacije o tome koliko pojedini ulaz nosi informacije vezane za neki drugi ulaz. Ovaj opisani mehanizam pozornosti se može vidjeti na slici 2.16.



Slika 2.16: Mehanizam pozornosti[31]

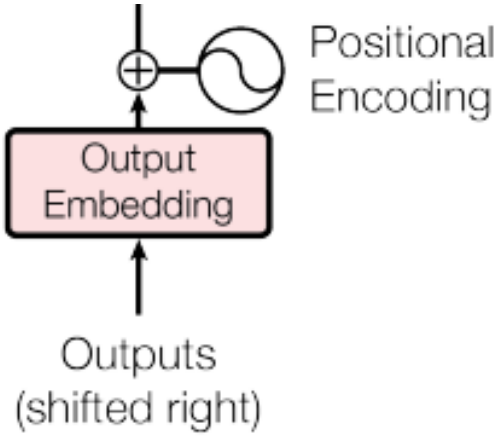
Mehanizam samopozornosti u koder sloju se provodi više puta, tj. u više glava gdje je svaka glava zadužena za ekstrakciju novih informacija o povezanosti ulaza.

Generirane vrijednosti iz sloja pozornosti za svaku glavu se konkateniraju u jednu reprezentaciju koja na kraju prolazi kroz linearni sloj te se prosljeđuje na drugi podsloj s unaprijednim potpuno povezanim slojevima (Slika 2.17).



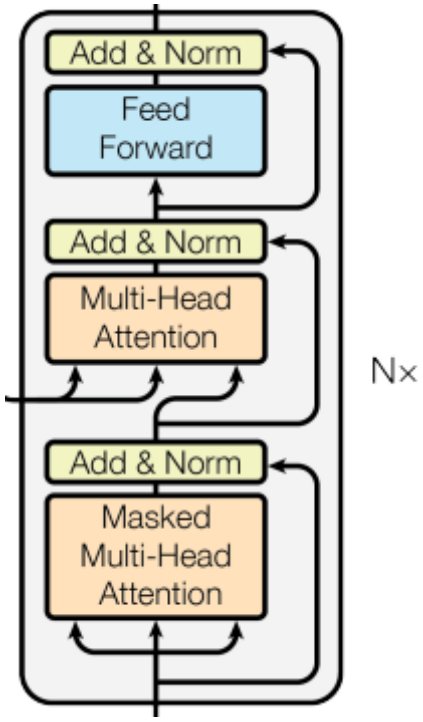
Slika 2.17: Mehanizam pozornosti s više glava[31]

Na sličan način se obrađuju i izlazni podaci, tj. željene vrijednosti, prilikom treniranja mreže. Prvo se podaci pretvore u ugrađene vektore pa se zatim u te vektore ugradi informacija o poziciji preko pozicijskog kodiranja (Slika 2.18).



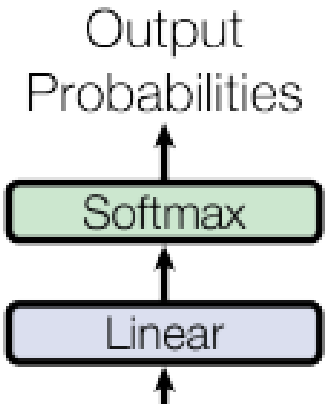
Slika 2.18: Priprema željenih vrijednosti za transformer neuronsku mrežu[31]

Dekoder slojeva također ima N te je njihova struktura sadrži iste podslojeve kao i koder sloj te uz to još jedan dodatni podsloj. Prvi podsloj obavlja gotovo istu samopozornost s više glava, gdje je jedina razlika to što se onemogućuje mreži postavljanje povezanosti u smjeru od prijašnjeg podatka prema budućem podatku, tj. jedino gdje će matrica sadržavati vrijednosti za mjere povezanosti je kod određenog podatka i svih podataka koji su došli prije njega. Način na koji se to provodi je da se pola matrice postavi na nul vrijednost (na svim mjestima gdje je indeks podatka od kojeg se gleda manji od indeksa podatka prema kojem se gleda). Drugi podsloj provodi mehanizam pozornosti s više glava između vrijednosti dobivenih iz koder sloja i vrijednosti dobivenih iz prvog podsloja dekoder sloja pa se sada određuju povezanosti između ulaza i izlaza. Nakon toga se na isti način provodi treći podsloj, tj. podsloj s unaprijednim potpuno povezanim slojevima. Također, na izlazu svakog podsloja se dodaju rezidualne veze i provodi normalizacija po sloju kao i u koder sloju. Arhitektura dekoder sloja se vidi na slici 2.19.



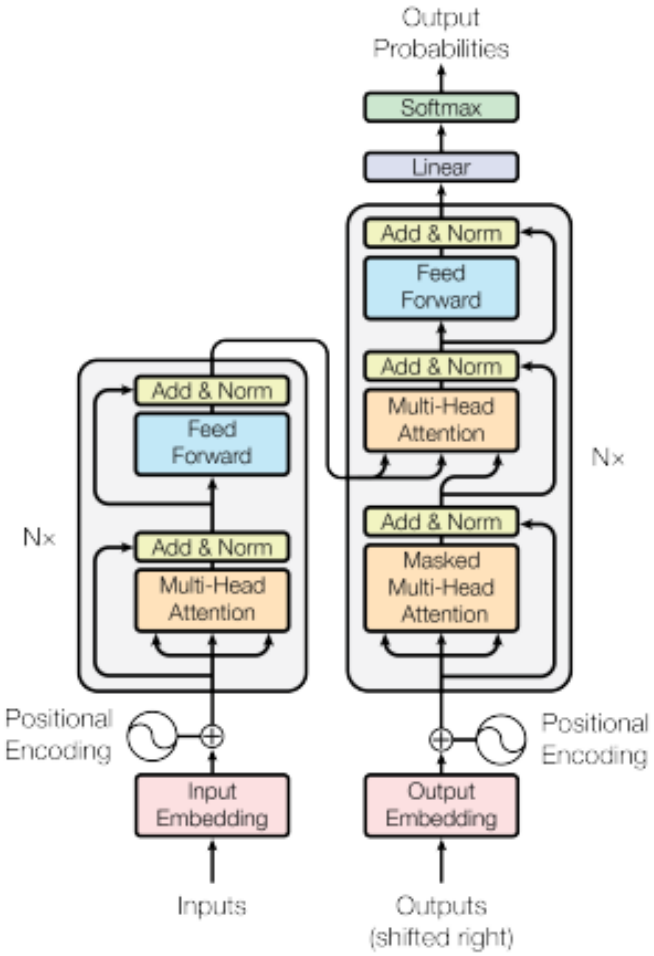
Slika 2.19: Dekoder sloj transformer neuronske mreže[31]

Izlaz iz posljednjeg dekoder sloja se provodi kroz linearni sloj i nakon njega kroz softmax sloj kako bi se dobila distribucija vjerojatnosti po definiranim izlaznim klasama (Slika 2.20.). Najvjerojatnija klasa predstavlja izlaz iz mreže te se prilikom predikcija novih neviđenih podataka taj izlaz donosi na ulaz dekoder slojeva sve dok mreže ne generira znak koji predstavlja kraj generiranja.



Slika 2.20: Izlaz iz transformer neuronske mreže[31]

Cijeli opisani proces rada transformer neuronske mreže se vidi na slici 2.21.



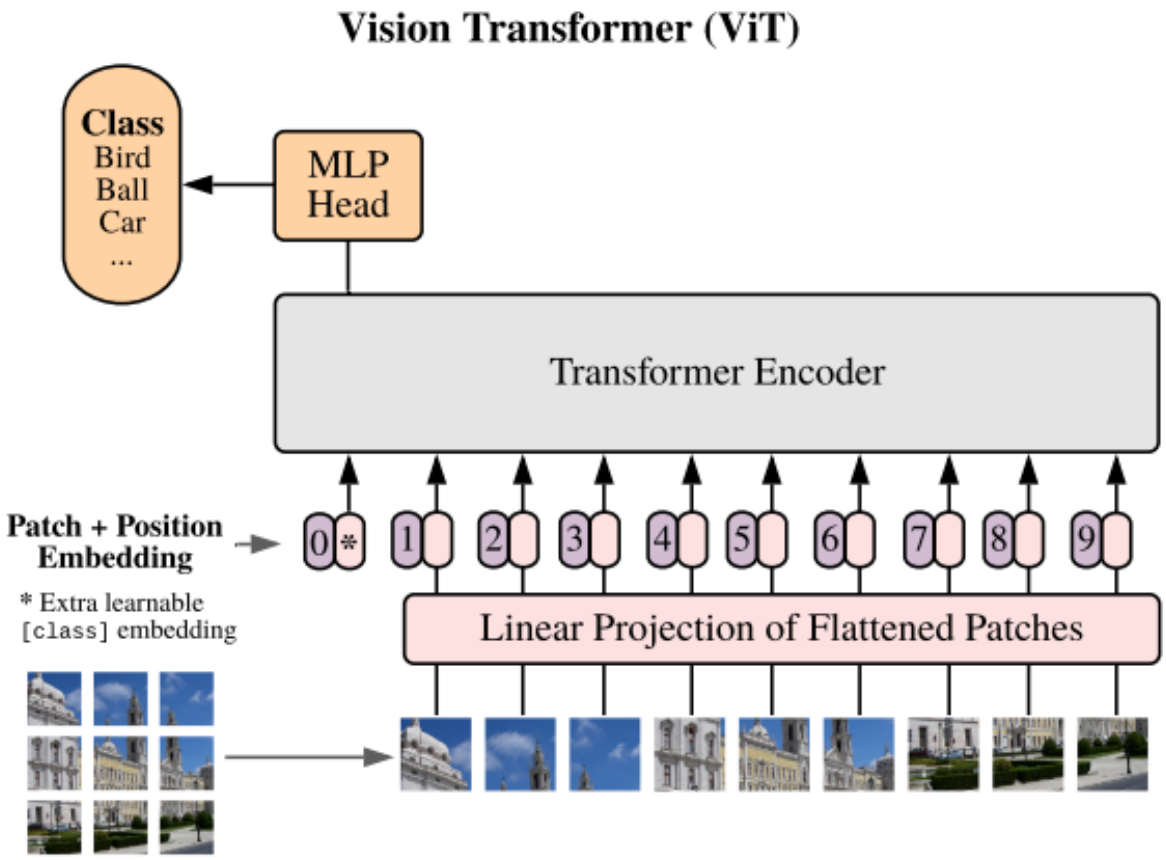
Slika 2.21: Primjer arhitekture transformer neuronske mreže[31]

Transformer neuronske mreže se danas široko primjenjuju u raznim područjima,

gdje se najčešće koriste u obradi prirodnog jezika (eng. *Natural Language Processing*), računalnom vidu, prepoznavanju zvuka te u sustavima za davanje preporuka (eng. *Recommender systems*).

Vizualni transformer

Vizualni transformer (eng. *Visual Transformer*) je varijanta transformer neuronske mreže koja je dizajnirana za rješavanje zadataka u području računalnog vida[15]. Ovaj model omogućava obradu slika sa transformerskom neuronskom mrežom koja je početno zamišljena za obradu prirodnog jezika. Kako bi to bilo moguće, slika se mora pretvoriti u oblik sekvencijalnih podataka koji će se zajedno poslati na ulaz mreže. Način na koji se to radi je da se slika razlomi u niz manjih nepreklapajućih dijelova iste veličine. Iz slike se izrežu manji dijelovi, npr. veličine 16x16 piksela, koji se postave na ulazni niz mreže. Te je dijelove potrebno pretvoriti u numeričke vektore pa se ti izrezani dijelovi izravnaju, tj. ako je jedan izrezani dio bio veličine 16x16, onda će izravnati vektor biti vektor s 256 vrijednosti. Obično vektori još prođu kroz linearni sloj s težinama koje su već istrenirane na sličnim zadacima te im se dodaje pozicijsko kodiranje. Pozicijsko kodiranje za slike se provodi tako da se svakom dijelu slike pridijeli indeks koji predstavlja redoslijed kojim su se izrezivali ti dijelovi, a dijelovi se izrezuju po principu šetajućeg prozora. Sami model prati sličnu arhitekturu već opisane transformer neuronske mreže, a razlika je u tome što se ne koristi dekoder sloj, već se izlaz koder sloja koristi izravno za predikciju. Mehanizam pozornosti u vizualnom transformeru pronalazi povezanosti između određenih dijelova slike. Budući da vizualni transformer u isto vrijeme može vidjeti sve dijelove slike, tj. cijelu sliku, može iz nje izvući globalni kontekst. Ovo je razlika s obzirom na standardni model koji koristi konvolucijske slojeve gdje se izvlači lokalni kontekst koji se prenosi kroz niz konvolucijskih slojeva kako bi se mogao izvući globalni kontekst. Vizualni transformer, kao i ostali modeli za prepoznavanje teksta, standardno koristi CTC (eng. *Connectionist Temporal Classification*) funkciju gubitka za ažuriranje težina. Ova funkcija gubitka se koristi za usklađivanje ulaznog niza sa željenim izlaznim nizom[17]. Kada govorimo o problemu prepoznavanja teksta na slici, onda ona određuje koji ulazni dio slike odgovara kojem izlaznom dijelu teksta, tj. ulazni dijelovi slike se usklađuju sa znakovima. U obzir se uzimaju sva moguća usklađivanja te računaju njihove vjerojatnosti. Na izlaz se prosljeđuje najvjerojatnije usklađivanje, a ono se kažnjava ako nije ispravno. Primjer pripreme slike za vizualni transformer i njegov princip rada se može vidjeti na slici 2.22.

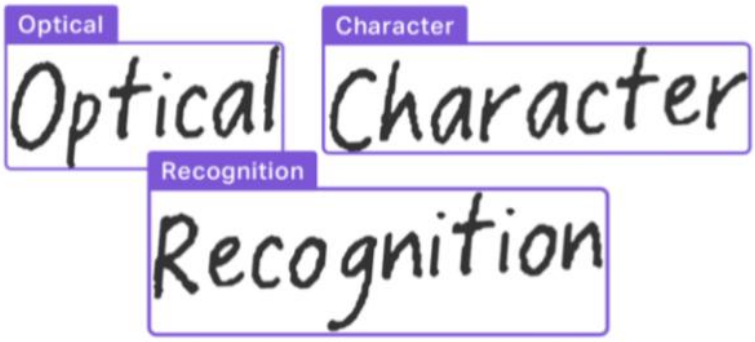


Slika 2.22: Primjer arhitekture vizualnog transformera[15]

Iako transformerske neuronske mreže provode složenije izračune od konvolucijskih slojeva, rad u kojem su uvedeni vizualni transformeri je uspio pokazati da takva mreža može u kraćem vremenu postići bolje rezultate od mreže temeljene na konvolucijskim slojevima na standardnim podatkovnim skupovima iz područja računalnog vida[15]. U nekim je slučajevima i dalje bolje koristiti konvolucijske slojeve za zadatke računalnog vida kao što je prepoznavanje teksta na slici, a primjer za to bi bio prepoznavanje teksta na slikama sa strukturirani oblikom teksta kao što je nekakav dokument u kojem se teksta nalazi u pravilnim redovima. Također, konvolucijske slojeve je dobro koristiti za manje zadatke gdje mreža koja je puno jednostavnija od transformer neuronske mreže može davati zadovoljavajuće rezultate.

2.2.5. Arhitektura sustava za optičko prepoznavanje znakova

Nakon što su definirani potrebni dijelovi sustava za optičko prepoznavanje znakova, mogu se spojiti u cjeloviti sustav. Kako bi se provelo optičko prepoznavanje znakova na slici, potrebno je prvo pretprocesirati sliku kako bi se ona pripremila za lokalizaciju samog teksta. Lokalizacijom se odrede dijelovi slike na kojima se nalazi tekst te se oni izrežu i postave kao zasebne slike nad kojima se napravi potrebno pretprocesiranje kako bi bile spremne za model. Kako bi se mogao trenirati model za optičko prepoznavanje znakova, potrebno je unaprijed pripremiti podatke, tj. potrebno je pronaći slike s tekstom na sebi, lokalizirati taj tekst te označiti sliku s ispravnom oznakom teksta. Kada se istrenira model, moguće je prepoznati tekst na novoj, modelu neviđenoj, slici tako da se prvo provede kroz model za lokalizaciju ako tekst nije lokaliziran pa onda kroz istrenirani model. Na slici 2.23 se može vidjeti primjer ispravnog rezultata cijelog opisanog procesa, od lokalizacije teksta do prepoznavanja samog teksta. Na slici je tekst lokaliziran ljubičastim pravokutnim okvirom, a rezultat modela za svaki okvir za optičko prepoznavanje je napisan iznad okvira.



Slika 2.23: Primjer ispravnog prepoznavanja teksta na slici[3]

2.3. Izazovi višealfabetnog prepoznavanja teksta

Kako bi sustav mogao ispravno izvršavati višealfabetno prepoznavanje teksta na slici, mora na pravilan način pristupiti izazovima koje ono nosi sa sobom. Prepoznavanje jednog alfabeta je samo po sebi složen zadatak, a kombinacija različitih alfabeta nadodaje nove komplikacije. Neki od izazova koji se pojavljuju su sljedeći:

Raznolikost jezika i alfabeta je velika, a svaki jezik i alfabet dolaze sa svojim jedinstvenim karakteristikama, tj. novim skupovima znakova i raznim stilovima pisanja. Ovo rezultira s povećanom složenošću problema, gdje model mora moći hvatati veći broj pravila i obrazaca u podacima u isto vrijeme.

Razi oblici znakova između alfabeta rezultiraju s varijacijama u obliku, veličini i stilu pisanja znakova. Također, različiti alfabeti mogu sadržavati slične znakove koji se mogu teško razlikovati. Ovaj problem je posebno istaknut ako se radi o tekstu koji je napisan rukom.

Ligature predstavljaju znakove koji su nastali spajanjem dvaju (ili više) znakova u jedan novi znak[33]. One se pojavljuju u velikom broju alfabeta, a česte su u istočnoazijskim i arapskim alfabetima, ali se pojavljuju i u latinici. Ovakve kombinacije znakova otežavaju modelu ispravnu segmentaciju na pojedinačne znakove te zahtijevaju složeniji proces obrade podataka kako bi bili spremni za treniranje modela.

Priprema podataka za treniranje i evaluaciju je složenija, a ne postoji velik broj javno dostupnih skupova podataka za ovaj problem. Kako bi se model mogao ispravno istrenirati, potrebno je napraviti veliki skup podataka s dovoljno primjera za svaki željeni alfabet.

2.4. Korištene tehnologije

Python sa svojim bibliotekama je postao glavni izbor za implementaciju sustava koji se baziraju na metodama strojnog učenja, pa se stoga i isključivo koristi za izgradnju sustava za optičko prepoznavanje znakova. Korištene su sljedeće Python biblioteke:

PIL (eng. *Python Imaging Library*) je biblioteka koja sadrži metode za obradu slika. PIL omogućuje prikazivanje, izmjenu i spremanje raznih formata slikovnih datoteka, od kojih su osnovni .jpg i .png. Od izmjena slika omogućuje obrezivanje (eng. *cutting*), promjenu veličine, promjenu boje, zamućivanje, izoštravanje, rotiranje i mnoge druge operacije.

XML biblioteka služi za obradu .xml (eng. *Extensible Markup Language*) datoteka. Ona omoguće manipulaciju, generiranje i spremanje .xml datoteka. Glavna potrebna funkcionalnost ovog modula je podmodul ElementTree koji iskorištava hijerarhijsku strukturu .xml datoteka tako da ju prikaže kao stablo u kojem su elementi datoteke prikazani kao čvorovi koji se mogu lako dohvatiti [14].

Torch biblioteka sadrži metode potrebne za izgradnju, treniranje i evaluaciju modela dubokog učenja. Ova biblioteka koristi svoj tip podatka koji se naziva tenzor. Tenzor je polje podataka željene dimenzije koje omogućuje efikasno izvođenje matematič-

kih operacija, a ta se polja koriste u svim strukturama podataka ove biblioteke. Torch omogućuje jednostavnu izgradnju arhitekture neuronske mreže te oslobađa korisnika od implementacije računanja gradijenata koje obavlja automatski kroz tenzore. Ova biblioteka je stekla veliku popularnost u zajednici koja se bavi dubokim učenjem zbog svoje učinkovitosti, jednostavnosti i široke podrške za treniranje neuronske mreže.

Torchvision biblioteka predstavlja proširenje Torch biblioteke za obradu metoda dubokog učenja i to u pogledu računalnog vida. Ova biblioteka omogućuje lakšu obradu slika, korištenje gotovih skupova podataka te unaprijed istreniranih modela za ekstrakciju značajki iz slika.

Trdg (eng. *TextRecognitionDataGenerator*) je biblioteka koja omogućuje generiranje umjetnih podataka za optičko prepoznavanje znakova. Kako bi ovo moglo biti moguće, potrebno je biblioteci pripremiti tekstualnu datoteku s riječima željenog jezika kao i razne fontove tog jezika, a koje će ona onda pretvarati u slike.

Arabic-reshaper je biblioteka koja se koristi za ispravno ispisivanje arapskih znakova i riječi. Ona omogućuje zapisivanje arapskih oznaka kako bi se te oznake mogle koristiti za treniranje modela u Python-u.

Hangul-jamo biblioteka omogućuje rastavljanje i sastavljanje korejskih ligatura. Korejski jezik kombinira osnovne znakove abecede u nove složenije znakove pa se ti složeniji znakovi rastavljaju na osnovne pomoću ove biblioteke s ciljem da modeli znaju od kojih se točno osnovnih znakova sastoji pojedina korejska riječ.

3. Priprema skupova podataka

U ovom poglavlju se opisuje način prikupljanja i pripreme podataka koji se koriste za treniranje i testiranje višealfabetnog sustava za optičko prepoznavanje znakova. Najprije je bilo potrebno odabrati alfabete za koje će se graditi sustav. Na tu odluku je utjecao broj javno dostupnih podatkovnih skupova za određeni alfabet i tako su odabrani sljedeći alfabeti: latinica (engleski jezik), arapski alfabet i korejski alfabet. Za svaki alfabet je potrebno pripremiti dva podatkovna skupa, a koji će služiti za treniranje i testiranje modela. Ustanovljen je datotečni oblik koji svaki skup podataka treba imati kako bi mogao biti učitan pomoću iste metode, a oblik je sljedeći: sve slike za taj podatkovni skup se nalaze unutar jedne mape, a naziv svake slike je indeks broja slike i njezin transkriptirani tekst. Primjer ovog oblika, gdje slika ima indeks "1005" i "world" kao napisani tekst, se može vidjet na slici 3.1.

Slika 3.1: Primjer oblika pripremljene slike

Budući da za sva tri alfabeta ne postoji javno dostupan skup podatka koji je dovoljno velik za potrebe treniranja modela, prikupljeni podatkovni skupovi se koriste za testiranje modela, a za treniranje se umjetno generiraju podatkovni skupovi. U slje-

dećim potpoglavljima se opisuje dohvaćanje i priprema javno dostupnih podatkovnih skupova za svaki od tri alfabeta kao i proces generiranja umjetnih podataka za svaki alfabet koji se koriste za treniranje modela.

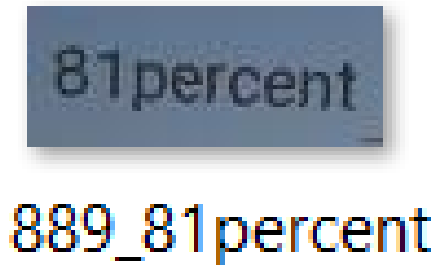
3.1. Priprema podatkovnih skupova za latinicu

Za latinicu je odabran engleski jezik, zbog svoje velike količine javno dostupnih podataka. Dohvaćen je ICDAR (eng. *International Conference on Document Analysis and Recognition*) 2015 podatkovni skup koji je napravljen za natjecanje u robusnom čitanju znakova (primjer na slici 3.1)[4]. ICDAR 2015 se sastoji od slika teksta uhvaćenih u prirodnim okruženjima te često to podrazumijeva izazovno okruženje poput lošeg osvjetljenja, loše perspektive, složene pozadine i zamućenosti. Pripremljeno je 1218 slika iz ovog skupa podatka za skup za testiranje. Nakon što se pripremila mapa s podacima kao s primjera sa slike 3.1, provela se analiza korištenog alfabeta za oznake. Analizom se točno odredio niz znakova koji će biti mogući izlazi modela. Takvih znakova je 62, od kojih je 26 malih i 26 velikih slova engleske abecede te 10 znakova za znamenke. Znakovi latinice su prikazani na slici 3.2. Za generiranje umjetnih podataka za treniranje je bilo potrebno pripremiti tekstualnu datoteku s engleskim riječima i mapu s različitim engleskim fontovima. Engleske riječi su prikupljene (eng. *scraped*) s raznih internetskih stranica te su obrađene tako da su obrisani svi znakovi koji nisu u definiranih 62 znaka i tako je ostalo 1117590 riječi. Prikupljeno je i 71 različitih datoteka s engleskim fontovima. Pomoću riječi i fontova je generirano 100000 primjera za treniranje, gdje slika 3.3 prikazuje jedan takav primjer s indeksom 889 i tekstom "81percent".

Slika 3.2: Latinični znakovi

3.2. Priprema podatkovnih skupova za arapski alfabet

Za arapski alfabet su prikupljeni podaci iz EvArEST (eng. *Everyday Arabic-English Scene Text dataset*) podatkovnog skupa, koji je skup arapskih i engleskih primjera za prepoznavanje teksta[19]. Ovaj podatkovni skup je dizajniran za prepoznavanje teksta u dvojezičnim arapsko-engleskim scenama, a sastoji se od slika uhvaćenih u prirodnim okruženjima. Riječi na slikama dolaze u raznim fontovima, orijentacijama,

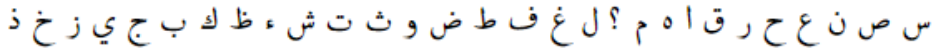


Slika 3.3: Primjer umjetno generirane slike za latinicu

veličinama i uvjetima osvjetljenja. Iz ovog podatkovnog skupa je pripremljeno 1035 slika sa arapskim tekstom za skup za testiranje(slika 3.4).

Slika 3.4: Primjer pripremljene testne slike za arapski alfabet

Na prikupljenim slikama se nalazi 40 različitih znakova, od kojih su 28 znakova osnovne arapske abecede, 2 dodatna arapska rečenična znaka te 10 znakova za znamenke. Znakovi arapskog alfabeta su prikazani na slici 3.5. S raznih arapskih internetskih stranica su dohvaćene jedinstvene arapske riječi, a one su zatim obrađene tako da su izbrisani oni znakovi koji se ne pojavljuju u 40 znakova iz testnog skupa i tako je pripremljeno 831780 različitih arapskih riječi te uz to je dohvaćeno 19 datoteka s različitim arapskim fontovima. Umjetno je generirano 100000 slika s arapskim tekstom za treniranje modela. Primjer generirane slike se može vidjeti na slici 3.6.



Slika 3.5: Znakovi arapskog alfabeta

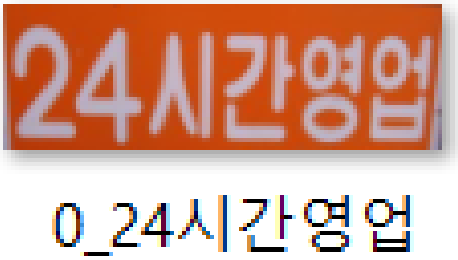
Na slici 3.4 se može uočiti da je tekst na slici napisan obrnutim redoslijedom nego u nazivu slike. Razlog ovoj pojavi se što se arapski tekst piše s desna na lijevo i tako ga obrađuje Python. Prilikom učitavanje ovakvih slika se arapski tekst prvo provodi kroz

Slika 3.6: Primjer umjetno generirane slike za arapski alfabet

metode biblioteke arabic-reshaper (definirane u poglavlju o korištenim tehnologijama) kako bi imao istu orijentaciju kao i na slici.

3.3. Priprema podatkovnog skupa za korejski alfabet

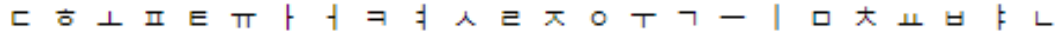
Slike s tekstom napisanim na korejskom alfabetu su dohvaćene iz KAIST (eng. *Korea Advanced Institute of Science and Technology*) podatkovnog skupa za prepoznavanje teksta. KAIST je istraživačko sveučilište smješteno u Daejeonu u Južnoj Koreji. Osnovno je 1971. godine, a poznato je po svojim doprinosima u STEM (eng. *science, technology, engineering, and mathematics*) disciplinama[22]. Ovaj podatkovni skup je kreirao tim istraživača, a on se sastoji od različitih slika uhvaćenih u prirodnim okruženjima. Tekst na slikama dolazi u raznim fontovima, veličinama, orijentacijama i stilovima. Iz podatkovnog skupa je pripremljeno 190 slika koje će se koristiti kao testni skup za korejski alfabet. Oznake za tekst na slikama su došle u .xml datotekama pa su ekstraktirane pomoću xml python biblioteke. Primjer pripremljene slike za korejski alfabet se može vidjeti na slici 3.7.



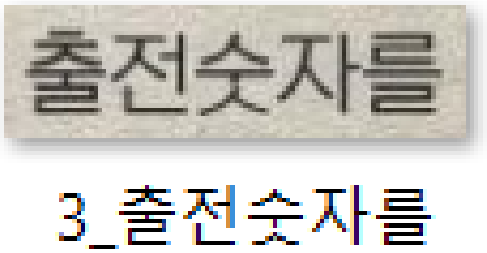
Slika 3.7: Primjer pripremljene testne slike za korejski alfabet

Na prikupljenim slikama se nalazi 34 jedinstvena znaka, od kojih je 24 znaka os-

novne korejske abecede (slika 3.8) te je 10 znakova za znamenke. Kao i za prethodna dva alfabeta, i za korejski alfabet su prikupljene jedinstvene riječi koje su obrađene tako da su obrisani znakovi koji se ne pojavljuju u prethodno navedenih 34 znaka. Tako je za korejski alfabet prikupljeno 8705600 različitih korejskih riječi te uz njih je prikupljena 21 datoteka s različitim korejskim fontovima. Pomoću teksta i fontova je umjetno generirano 100000 primjera korejskog alfabeta za treniranja modela. Primjer jedne umjetno generirane slike s tekstom se može vidjeti na slici 3.9.



Slika 3.8: Znakovi korejskog alfabeta



Slika 3.9: Primjer umjetno generirane slike za korejski alfabet

Na slikama 3.7 i 3.9 se korejski tekst nalazi u obliku ligatura koje je potrebno rastaviti kako bi model mogao točno znati koji znakovi se pojavljuju u tekstu. Rastavljanje ligatura se provodi pomoću hangul-jamo Python biblioteke.

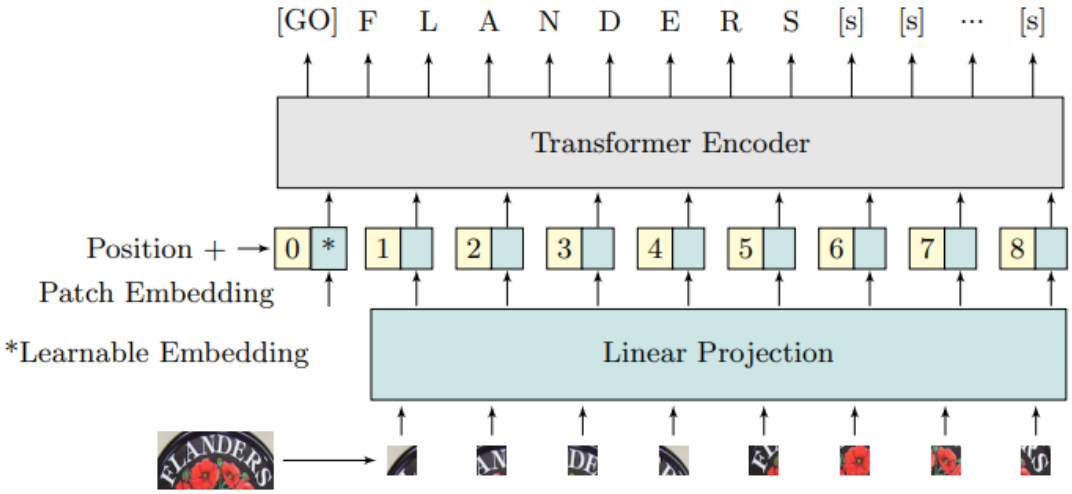
4. Programsko ostvarenje sustava za optičko prepoznavanje znakova i rezultati

U ovom poglavlju se opisuje programsko ostvarenje sustava za optičko prepoznavanje znakova i rezultati koje je taj sustav postigao. Sustav prati model arhitekture iz poglavlja 2.2.5. U sustavu za optičko prepoznavanje znakova je prvo je potrebno provesti lokalizaciju teksta na slici, a taj korak se preskače u ovom radu jer je tekst na slikama prikupljenim u poglavlju 3 već lokaliziran. U potpoglavlju 4.1. se opisuje korišteni model za prepoznavanje teksta, u potpoglavlju 4.2. se navode implementacijski detalji vezani za treniranje modela te se u potpoglavlju 4.3. iznose rezultati koje je postigao ovaj sustav.

4.1. Detalji arhitekture korištenog vizualnog transformera

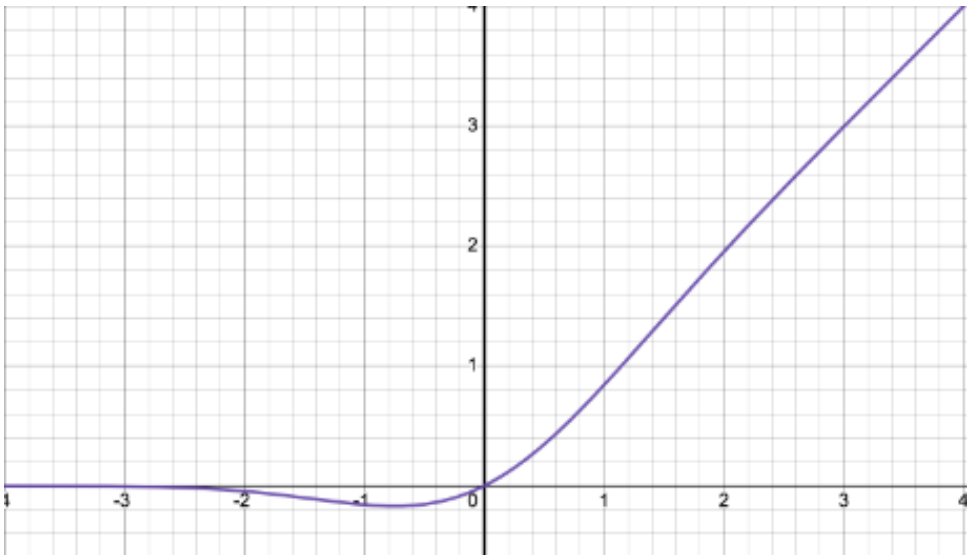
U ovom radu je korišten vizualni transformer kao model za prepoznavanje teksta na slici. Arhitektura tog modela se može vidjeti na slici 4.1. Slika koja se pojavljuje na ulazu je dimenzija C x H x W, gdje C predstavlja broj ulaznih kanala slike, a H i W predstavljaju visinu i širinu slike. Kao što je opisano u poglavlju 2.2.4., slika se prije vizulanog transformera dijeli na manje nepreklapajuće dijelove veličine C x P x P, gdje P predstavlja veličinu kvadratnog dijela koji se uzima iz slike. Ti se dijelovi onda izravnaju u vektor veličine $C * P^2$ koji se još provodi kroz linearni sloj kako bi se dobio vektor dimenzionalnosti D, a D predstavlja veličinu koja mreža zahtijeva na ulazu. Svim se vektorima još dodaje pozicijsko kodiranje te se svi zajedno konkateniraju u jednu reprezentaciju ulaznog niza podataka za mrežu.

Kada su podaci pretvoreni u odgovarajući vektorski oblik, prosljeđuju se u prvi ko-



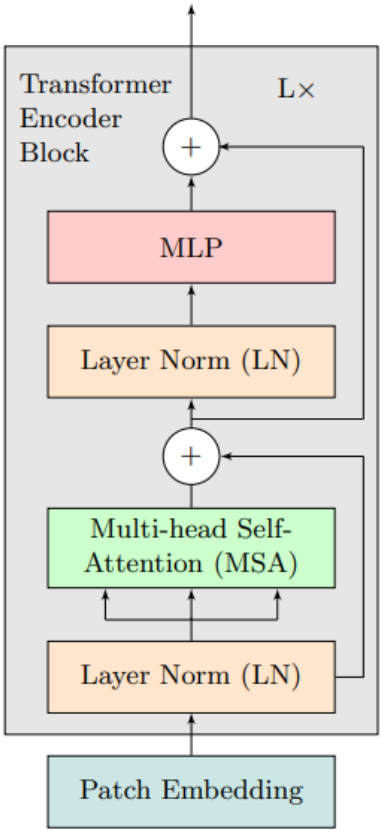
Slika 4.1: Korišteni vizualni transformer[12]

der sloj. Arhitektura korištenog koder sloja se vidi na slici 4.3. Prvo se ulazni podaci normaliziraju pa se prosljeđuju u podsloj koji provodi samopozornost s više glava koji određuje povezanosti između ulaznih vektora. Normalizirani vektori prije i poslije pozornosti se zbrajaju i ponovno normaliziraju te se takvi provode kroz podsloj potpuno povezanih slojeva. Podsloj potpuno povezanih slojeva se sastoji od dva sloja između kojih se koristi GELU (eng. *Gaussian Error Linear Units*) aktivacijska funkcija, a čiji je izgled prikazan na slici 4.2.



Slika 4.2: GELU aktivacijska funkcija[21]

Izlaz iz potpuno povezanih slojeva se zbraja s izlazom iz prvog podsloja koji provodi pozornost u konačan izlaz jednog od L broja koder slojeva.



Slika 4.3: Arhitektura koder sloja[12]

4.2. Implementacijski detalji vezani za treniranje modela

U ovom poglavlju se navode vrijednosti parametara i hiperparametara koji su bili korištenih za postavljanje i treniranje modela. Slike iz kojih se želi ekstraktirati tekst se prvo postave na veličinu 224 x 224 i postavi im se broj kanala na 1 (slike su pretvorene u crno-bijeli oblik), a nakon čega se iz njih uzimaju manji dijelovi s P veličinom od 16, tj. svaka slika se podijelila na manje nepreklapajuće dijelove veličine 16 x 16. Svaki manji dio se izravnao u vektor koji je još prošao kroz linearni sloj gdje postiže dimenzionalnost od 192. Sami vizualni transformer se sastoji od 12 koder slojeva u kojima se pozornost obavlja s 3 glave. Na izlazu modela je korištena CTC funkcija gubitka. Prilikom treniranja modela je bilo potrebno postaviti određene hiperparametre kao što su: veličina grupe za treniranje (eng. *Batch size*), stopa učenja (eng. *Learning rate*), optimizator, funkcija gubitka te mjera skraćivanja gradijenta (eng. *Gradient clipping*). Veličina grupe za treniranje je odabrana s obzirom na dostupnu memoriju i postavljena

na 32. Za stopu učenja je uzeta vrijednost od $3*10^{-4}$. Korišten je Adam optimizator s beta vrijednostima od 0.9 i 0.95. Inicijalne težine modela su generirane Kaiming inicijalizacijom [20] te mjera skraćivanja gradijenata je postavljena na 5.

4.3. Rezultati

U ovom poglavlju se navode rezultati treniranih modela. Trenirano je 3 modela: model za višealfabetno prepoznavanje teksta sa sva tri alfabeta (la+ar+ko), model za arapski alfabet (ar), model za korejski alfabet (ko), a za model s latinicom (la) je korišten već istrenirani javno dostupni model pod nazivom `vitstr_small_patch16_224_aug`. Jedina razlika u modelima je u tome što imaju različiti broj izlaznih klasa, tj. rade prepoznavanje različitog broja znakova, a to ovisi o broju znakova alfabeta. Model za sva tri alfabeta ima 116 izlaznih klasa, model za latinicu ima 96 izlaznih klasa, model za arapski alfabet ima 40 izlaznih klasa te model za korejski alfabet ima 34 izlaznih klasa. U podatkovnom skupu za latinicu iz poglavlja 3.1, izlaznih klasa je 62, a u dohvaćenom modelu je 96, ali svih 62 znakova se nalazi u tih 96 pa je dohvaćeni model moguće primjeniti na skup za testiranje. Trenirani modeli su ispitani na skupu za testiranje i to koristeći točnost, tj. mjeru preklapanja između predikcije i stvarnog teksta, kao metriku. Tablica 4.1 sadrži rezultate modela.

Tablica 4.1: Tablica rezultata

model	**točnost (%)**	**vrijeme treniranja(sati)**
la+ar+ko	45.027	14.076
la	70.115	-
ar	95.169	3.943
ko	67.368	7.200

U tablici se može vidjeti da najmanju točnost postiže model za sva tri alfabeta. Ovaj se model najduže trenirao, a neki od mogućih razloga lošije točnosti su: malen skup za treniranje, sličnosti u oblicima znakova različitih alfabeta te velika razlika između podataka za treniranje i podataka za testiranje jer su prvi umjetno generirani. Nad umjetno generiranim podacima su primijenjene metode pretprocesiranja (dodavanje šuma i rotiranje slike) kako bi model mogao bolje generalizirati nad stvarnim podacima, ali i dalje će postojati razlike između umjetnih i stvarnih podataka koje model neće vidjeti pa neće moći niti naučiti. Najveću mjeru točnosti od 95.169% je

postigao model koji je treniran samo na arapskom alfabetu. Skup za testiranje ovog alfabeta je najkvalitetniji od ova tri alfabeta, tj. ne sadrži puno loše prikazanog teksta. Modeli koji provode prepoznavanje teksta za jedan alfabet se ne mogu sami po sebi koristiti kao sustav za višealfabetno prepoznavanje teksta jer bi se prvo trebala izvući informacija koja govori o kojem se alfabetu radi pa se tek onda može proslijediti ulaznu sliku na odgovarajući model. Dakle, potreban je dodatan model koji će prvo odrediti alfabet prisutan na slici da bi se modeli s jednim alfabetom mogli spojiti u sustav višealfabetnog prepoznavanja teksta na slici.

5. Zaključak

Na kraju su istrenirani modeli uspjeli pokazati da je izgradnja modela koji pokušava naučiti više alfabeta u isto vrijeme jako složen zadatak koji zahtijeva veliku količinu resursa. Modeli koji uče jedan alfabet se lakše uče jer nema preklapanja između stilova znakova, a ono može gurati model u krivi smjer. Iako su modeli s jednim alfabetom uspjeli postići bolje individualne rezultate, ti se modeli ne mogu sami po sebi koristiti za višealfabetno prepoznavanje znakova. Kako bi se izgradio sustav od modela treniranih na pojedinačnom alfabetu, potrebno je prvo odrediti jezik koji je u pitanju pa zatim onda proslijediti sliku odgovarajućem modelu, a za to bi trebalo izgraditi poseban model. Ovim radom se iznose osnove izgradnje višealfabetnog sustava za prepoznavanje znakova, ali je tu još dosta prostora za daljnja poboljšanja. Neka od mogućih poboljšanja su: generirati veći skup za treniranje, koristiti skup za treniranje sličniji skupu za testiranje, primijeniti veći skup metoda za pretprocesiranje, prikupiti veći skup riječi za svaki alfabet te kreirati skupove podataka za nove alfabete te istrenirani model i za njih.

LITERATURA

[1] ReLU function - AILEPHANT — ailephant.com. https://ailephant.com/glossary/relu-function/. [Accessed 02-Jun-2023].

[2] Current State of OCR in 2023: Is it dead or a solved problem? — research.aimultiple.com. https://research.aimultiple.com/ocr-technology/. [Accessed 02-Jun-2023].

[3] — edenai.co. https://www.edenai.co/post/optical-character-recognition-ocr-which-solution-to-choose. [Accessed 02-Jun-2023].

[4] Competitions | ICDAR 2015 — iapr.org. https://iapr.org/archives/icdar2015/index.html%3Fp=254.html. [Accessed 03-Jun-2023].

[5] Deep Learning Based OCR for Text in the Wild — nanonets.com. https://nanonets.com/blog/deep-learning-ocr/. [Accessed 02-Jun-2023].

[6] Papers with Code - Tanh Activation Explained — paperswithcode.com. https://paperswithcode.com/method/tanh-activation. [Accessed 02-Jun-2023].

[7] Bidirectional recurrent neural networks - Wikipedia — en.wikipedia.org. https://en.wikipedia.org/wiki/Bidirectional_recurrent_neural_networks, . [Accessed 02-Jun-2023].

[8] Računalni vid – Wikipedija — hr.wikipedia.org. https://hr.wikipedia.org/wiki/Ra%C4%8Dunalni_vid, . [Accessed 02-Jun-2023].

[9] Recurrent neural network - Wikipedia — en.wikipedia.org. https://en.wikipedia.org/wiki/Recurrent_neural_network, . [Accessed 02-Jun-2023].

[10] Scene text - Wikipedia — en.wikipedia.org. https://en.wikipedia.org/wiki/Scene_text,. [Accessed 02-Jun-2023].

[11] Pratik Ahamed, Soumyadeep Kundu, Tauseef Khan, Vikrant Bhateja, Ram Sarkar, i Ayatullah Mollah. Handwritten arabic numerals recognition using convolutional neural network. *Journal of Ambient Intelligence and Humanized Computing*, 11, 11 2020. doi: 10.1007/s12652-020-01901-7.

[12] Rowel Atienza. Vision transformer for fast and efficient scene text recognition, 2021.

[13] Chandra Churh Chatterjee. An Approach Towards Convolutional Recurrent Neural Networks — towardsdatascience.com. https://towardsdatascience.com/an-approach-towards-convolutional-recurrent-neural-networks-a2e6c [Accessed 02-Jun-2023].

[14] cpython. xml.etree.ElementTree — The ElementTree XML API — docs.python.org. https://docs.python.org/3/library/xml.etree.elementtree.html. [Accessed 03-Jun-2023].

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, i Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[16] Anirudha Ghosh, A. Sufian, Farhana Sultana, Amlan Chakrabarti, i Debashis De. *Fundamental Concepts of Convolutional Neural Network*, stranice 519–567. 01 2020. ISBN 978-3-030-32643-2. doi: 10.1007/978-3-030-32644-9_36.

[17] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, i Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):855–868, 2009. doi: 10.1109/TPAMI.2008.137.

[18] Alex Graves, Navdeep Jaitly, i Abdel rahman Mohamed. Hybrid speech recognition with deep bidirectional lstm. *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, stranice 273–278, 2013.

[19] Heba Hassan, Ahmed El-Mahdy, i Mohamed E. Hussein. Arabic scene text recognition in the deep learning era: Analysis on a novel dataset. *IEEE Access*, 9: 107046–107058, 2021. doi: 10.1109/ACCESS.2021.3100717.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015.

[21] Dan Hendrycks i Kevin Gimpel. Gaussian error linear units (gelus), 2023.

[22] KAIST. KAIST — kaist.ac.kr. https://www.kaist.ac.kr/en/. [Accessed 04-Jun-2023].

[23] Andrej Karpathy. The Unreasonable Effectiveness of Recurrent Neural Networks — karpathy.github.io. http://karpathy.github.io/2015/05/21/rnn-effectiveness/. [Accessed 02-Jun-2023].

[24] Arvind N. Text Detection Using CRAFT Text Detector — analyticsvidhya.com. https://www.analyticsvidhya.com/blog/2022/06/text-detection-using-craft-text-detector/. [Accessed 02-Jun-2023].

[25] Amber Pariona. The world's most popular writing scripts, Oct 2019. URL https://www.worldatlas.com/articles/the-world-s-most-popular-writing-scripts.html.

[26] Sinisa Segvic. Duboko ux10D;enje — zemris.fer.hr. http://www.zemris.fer.hr/~ssegvic/du/. [Accessed 02-Jun-2023].

[27] SAGAR SHARMA. Activation Functions in Neural Networks — towardsdatascience.com. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. [Accessed 02-Jun-2023].

[28] Baoguang Shi, Xiang Bai, i Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition, 2015.

[29] Željko Trbušić. URL https://hrcak.srce.hr/file/352786.

[30] Savvas Varsamopoulos, Koen Bertels, i Carmen Almudever. Designing neural network based decoders for surface codes, 11 2018.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, i Illia Polosukhin. Attention is all you need, 2017.

[32] Chi-Feng Wang. The Vanishing Gradient Problem — towardsdatascience.com. https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484. [Accessed 02-Jun-2023].

[33] Leksikografski zavod Miroslav Krleža. ligatura | Hrvatska enciklopedija — enciklopedija.hr. https://www.enciklopedija.hr/natuknica.aspx?id=36456. [Accessed 03-Jun-2023].