Seminarska naloga modul J

Univerza v Ljubljani
Fakulteta za elektrotehniko

Uporaba nevronskih mrež za estimacijo cene proizvodnje

Jakob Jenko

januar 2022

Kazalo

Uvod

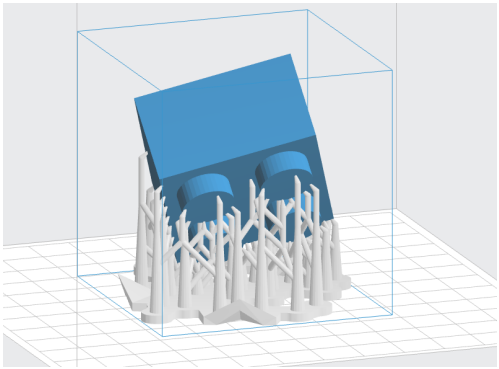
Malo proizvodnja industrija se sooča s problemom estimacije cene proizvodnje izdelka. Za razliko od velike pozivodnje, kjer se izdelujejo izdelki v velikih količinah male proizvodnje proizvajajo, manjše serije do nekaj 1000 kosov. V zadnjih letih se je 3D tiskanje uveljavilo kot ultra malo proizvodnja, kjer za potrebe prototipiranja natisnejo le nekaj 10 kosov. V primeru velike proizvodnje, kjer se kos proizvaja več mesecov, je estimacija cene zanemarljiv del. Če nam ocena cene vzame 15 minut, tisk in priprava delavne površine pa 1 uro, postane estimacija cene del stroška. Rešitev tega problema bi bila avtomatizacija estimacije cene z nevronskimi mrežami, kjer bi mreža iz podatkov 3D modela objekta predvidila ceno proizvodnje. Avtomatizacija bi prav tako omogočala predikcijo cene kot spletno storitev. Stranka bi na spletni strani proizvajalca naložila 3D objekt, ki bi ga želela natistniti, okvino ceno pa bi izvedela v nekaj sekundah.

1 Cena izdelave

Cena izdelave 3D izdelka je sestavljena iz več različno oteženih delov. Izdelava izdelka gre skozi 3 faze, ki jih bom predstavil v naslednjih poglavjih.

1.1 Priprava

Pred tiskom porabimo čas za pripravo 3D modela v rezalniku (angl. slicer), kjer moramo model na tiskalno posteljo postaviti tako, da ob tisku potrebujemo najmanj podpor. Podpora pri 3D tisku je dodaten material, ki ga moramo dodati pod dele modela, ki visijo in bi se brez podpor povesili. Prav tako moramo ustezno nastaviti parametre, da bo model dosegal dane specifikacije. Na koncu sledi še generiranje G-kode. Ponavadi je čas generacije zanemarljiv, v primeru velikih in kompleksih modelov z veliko točnostjo pa lahko traja tudi do 1 ure. Če zanemarimo čas generiranja, lahko rečemo, da je ta del stroška konstanten in neodvisen od modela.



Slika 1: Podpore vir: https://commons.wikimedia.org/wiki/File:Supports_in_3D_printing.png

1.2 Tisk

Pri tisku je cena sestavljena iz cene materiala in časa tiska. Tu je najpomebnejši del volumen modela, oziroma dolžina porabljenega filamenta. Bolj kot zapolnimo model, daljši čas tiska bo. Zelo velik vpliv imajo podpore, več kot jih je, daljši je čas tiskanja in več je porabljenega materiala. V tem delu je cena zelo odvisna od oblike modela.

1.3 Dodelava

Dodatno ceno dodaja dodelava, kjer moramo model odstraniti iz tiskalne površine in odstraniti ves dodaten material. Dodelava je sestavljena iz dela cene, ki je konstanten in iz dela, ki je odvisen od količine podpor.

1.4 Ostalo

Cena je prav tako odvisna od izbire materiala, cene opreme, delavne sile in marže. Ta del cene je konstanten in neodvisen od modela.

1.5 Enačba

Konča enačba, kjer so parametri

$$t = \text{čas } tiskanja$$
$$t_p = \text{čas } tiskanja\ podpor$$
$$m = masa\ filamenta$$
$$m_p = masa\ filamenta\ proabljenega\ za\ podpore$$
$$c_f = cena\ filamenta$$
$$c_{ds} = cena\ delavne\ sile$$
$$c_{\text{tisk}} = cena\ \text{tiskanja}$$
$$c = končna\ cena$$
$$k_{marža} = faktor\ marže$$

izračinamo ceno materiala

$$c_m = \left(m + m_p\right) c_f$$

ceno tiskanja na podlagi časa in cene tiska

$$c_t = (\ t\ + t_p)\ c_{\text{tisk}}$$

ceno dodelave na podlagi količine uporabljenih podpor

$$c_d = 2 * (m_p\ c_f\ + t_p\ c_{ds}\ )$$

ceno priprave delavne površine

$$c_p = k_{prip}\ c_{ds}$$

cena je seštevek vseh prispevkov

$$c = c_m + c_t + c_d + c_p$$

na koncu prištejemo še marž

$$c = c + k_{marža} * c$$

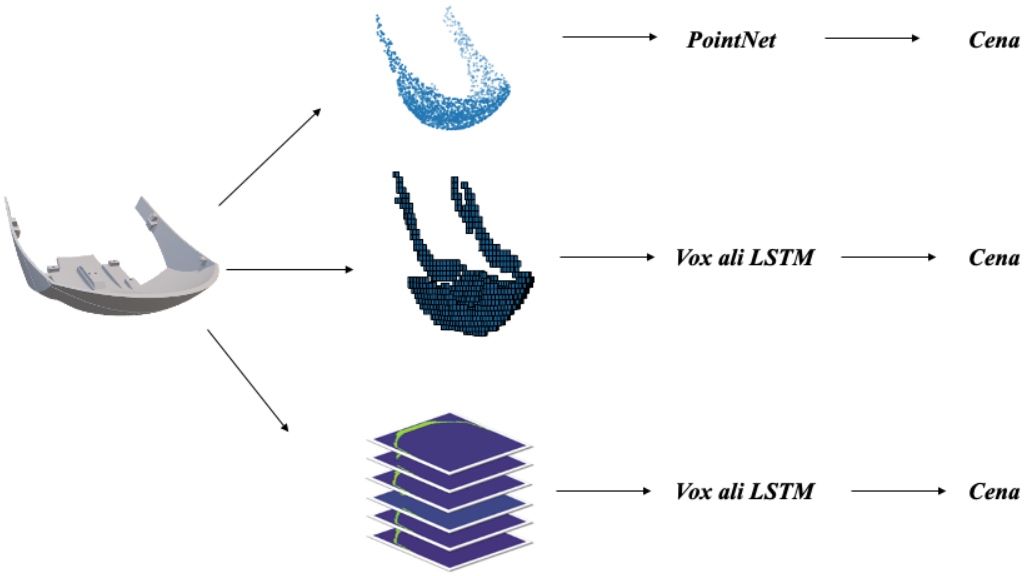
Enačbo sem sestavil s pomočjo članka [1]. S pomočjo spletne strani [2] sem preveril smiselnost in ujemanje cen.

2 Pregled trenutnega dela

V seminarski nalogi sem se v veliki meri zgledoval po članku [3], kjer sta avtorja sestavila model, ki zelo dobro oceni ceno CNC izdelave objekta. Avtorja sta primerjala dva tipa načina zapisa podatkov point cloud in voksle. Avtorica je v delu [1] analizirala, iz katerih delov je sestavljena cena in kakšen je vpliv posameznega dela na končno ceno. Avtorji v so delu [4] uporabili podatkovni pristop k predikciji cene s pomočjo 17 značilk, pridobljenih iz G-kode modela.

3 Cilji

Predvideval bom ceno 3D modela s pomočjo treh različnih formatov zapisa in treh različnih nevronskih mrež. Cilj je izdelati čim boljši model in ugotoviti, katere značilke največ prispevajo h kvalitetnemu modelu.



Slika 2: Ideja izvedbe

4 Metodologija

4.1 Podatki

Primernih podatkovnih setov nisem našel. Na spletu obstaja veliko portalov, kjer si skupnost za 3D tiskanje izmenjuje modele. Najbolj razširjen je Thingsverse [5], iz katerega sem pridobil 195 modelov različnih dimezij in velikosti.

4.1.1 Stuktura podatkov

En vzorec podatkov bo vseboval 3D model v formatu .stl, čas tiskanja brez podpor, čas tiskanja s podporami v sekundah, volumen filamenta brez podpor in volumen filamenta s podporami v mm^3, volumen in površina 3D modela.

4.1.2 Generiranje podatkov

Zgoraj omenjenim modelom moramo določiti čas tiskanja in količino filamenta. To storimo z rezalnikom oziroma »slicerjem«, to je progam, ki ga uporabimo za generiranje G-kode. Stranski produkt G-kode sta zgoraj omenjena parametra: čas tiskanja in količina porabljenega filamenta. Za ta postopek sem uporabil CuraEngine [6], ki je jedro programa Cura. Postopek rezanja poženemo enkrat s podporami, enkrat pa brez. To nam omogoča izračun časa tiskanja podpor in porabljenga filamenta za tiskanje podopor. Za izračun volumna iz površine modela je bila uporabljena python knjižnica trimesh [7].

Preden model pošljemo v rezalnik, ga moramo pravilno obrniti. Za iskanje optimalne orientacije z najmanj podporami je bil uporabljen program Tweaker-3 [8].

Pri generiranju sem nastavil osnovne parametre kot recimo mejo za podpore 70 °, z velikostjo šobe 0.4 mm, višino koraka 0.2mm in 25 % polnilo. To so najbolj pogoste in univerzalne nastavitve.

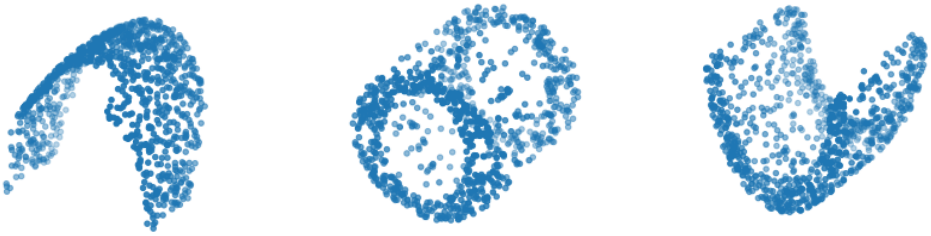
4.1.3 Stereolitografija (STL)

STL ali stereolitografija je način zapisa podatkov, kjer model predstavimo s trikotniki. Trikotniki so definirani z lokacijo in normalo, ki jih lahko skupaj sestavimo v 3D model. STL je ena izmed najbolj pogosto uporabljenih zapisov 3D objektov, saj ima zelo dobro razmerje med velikostjo datoteke in natačnostjo reprezetacije. Problem STL je, da jo moramo trasformirati, preden jo lahko uporabimo z nevronskimi mrežami.

Slika 3: Predstavitev 3D objekta s STL

4.1.4 Point cloud

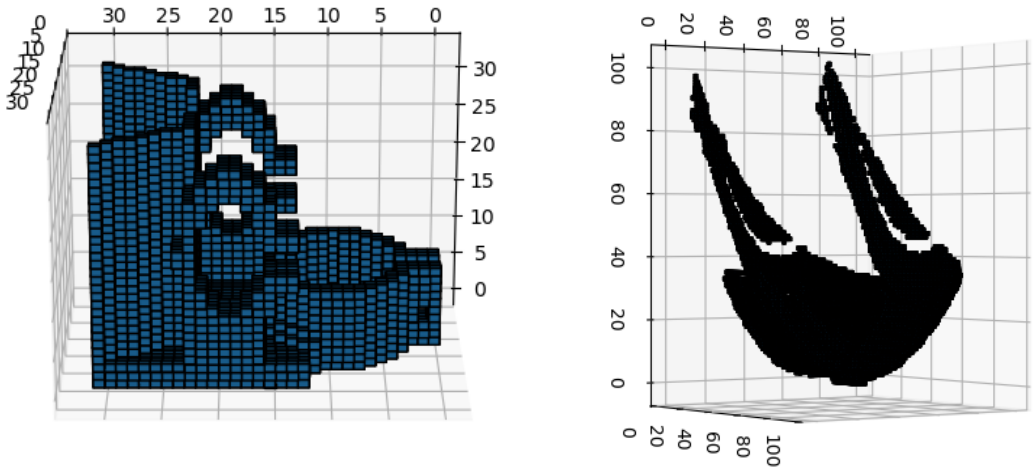
Point cloud zapis datotek se pogosto uporablja pri segmentaciji ali klasifikaciji objetkov. uporabljajo ga tudi algoritmi samo vozečih avtomobilov, ki uporabljajo Li-dar sistem. Problem 3D objektov je, da velikost datotek zelo hito narašča z velikostjo in kompleksostjo objektov. Ta problem naslovi point cloud, ki je zelo prostorsko učinkovit, vendar poda manj informacij in značilk kot ostali zapisi. Za trasfomracijo STL datotek v point cloud sem uprabil knjižnico Trimesh [7]. Modeli so predstavljeni z 4096 točkami.



Slika 4: Predsavitev objektov s point cloud

4.1.5 Voksli (INT)

Voksli so oblika zapisa, kjer za vsako koordinato x,y in z pove, ali se tam nahaja objekt ali ne. Pri tem se uporablja vrednosti 0 in 1, zapisane s celimi števili. V tem smislu so zelo podobni visoko kontrastnim črno-belim slikam, le da imajo dodatno dimenzijo. V primerjavi s point cloud ali STL so voksli prostorsko zelo potratni. Dobra lastnost je, da jih ni potrebno trasformirati, preden se jih uporabi v nevronskih mrežah. Za trasformacijo STL datotek v voksle sem uporabil program [9].



Slika 5: Predstavitev objetkov z voksli

4.1.6 Decimalni voksli (FLOAT)

V poglavju 4.1.7 opisujem avgmentacijo podatkov tam z interpolacijo izračunam nove pozicije vokslov. Zaradi celoštevilskega zapisa med 0 in 1 nastane problem pri rotaciji in interpolaciji. Na nekaterih mestih moram vrednost zaokrožiti in posledično izgubim veliko značilnosti na našem modelu. Problem lahko vidimo na sliki 8.

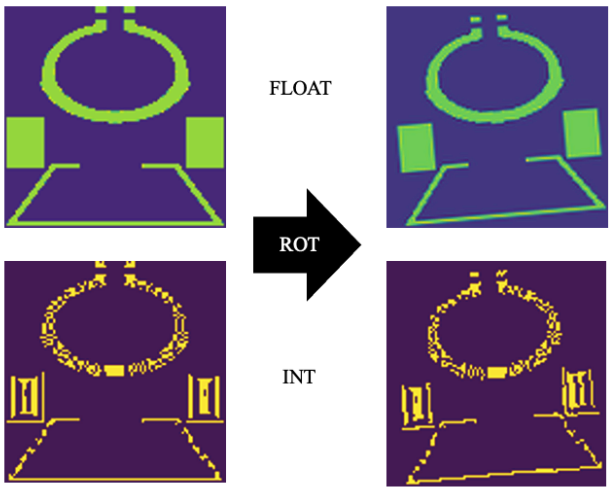
Posledično sem se odločil uporabiti 32 bitni decimalni zapis. Tako lahko ohranim značilke, ki so lahko potencialno pomembne in imajo veliko večjo ločljivost. Zaradi decimalnega zapisa nimam več vokslov, temveč piksle, ki so naloženi drug za drugim. Dimenzije slik ostajajo iste kot prej, le da imamo sedaj 100 slik dimenzije 100x100, kjer so slike razrezane po Z osi.

Slika 6: Sklad slik z decimalnim zapisom

4.1.7 Avgmentacija podatkov

Na strani point clouda lahko avgmentiramo podatke z naključnim premiki točk. Na strani vokslov avgmentacija podatkov poteka podobno kot pri 2D slikah. To delamo tako, da sliko premaknemo levo ali desno, rotiramo ali zrcalimo. V mojem primeru sem uporabil le rotacijo okoli Z osi, saj samo tako ostaja število podpor konstatno. Rotacija okoli x in y osi bi v praksi spremenila količino podpor, kar zmedlo mrežo. Podatke sem augmentiral 5 krat. Model sem zavrtel od 0 do 330 stopinj s korakom 60 stopinj. Set za učenje je imel tako malo manj kot 1000 vzorcev.

Slika 7: Rotacija prijemala za 360° okoli z osi (slika je 50 plastnica od 100)



Slika 8: Zgoraj rotacija decimalnega zapisa, spodaj rotacija vokslov.

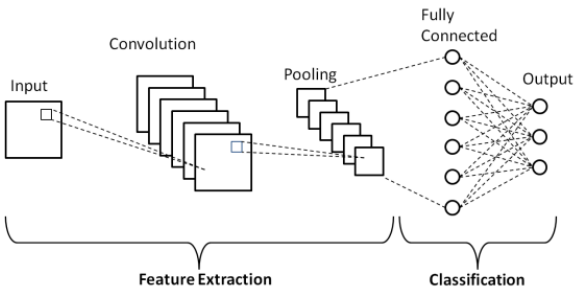
4.2 Strojno učenje

Eno izmed glavnih orodij pri strojnem učenju so konvolucijske nevrosnke mreže. Te pogosto uporabljamo pri klasifikaciji časovnih vrst, še boljše rezultate pa prikazujejo na slikah.

4.2.1 Konvolucijske nevronske mreže

Na spodnji sliki lahko vidimo, da je CNN sestavljen iz dveh glavnih delov:

- ekstrakcije značilk (angl. feature extraction) in
- klasifikacije (angl. classification).

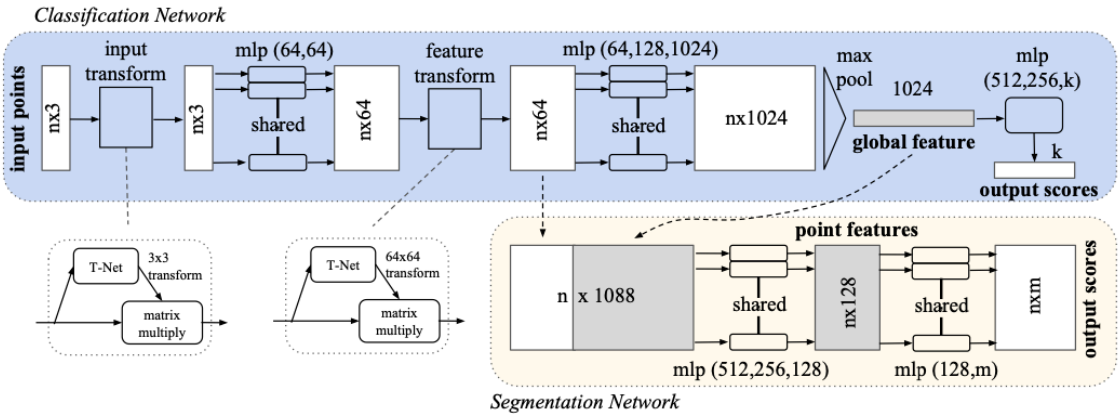


Slika 9: Konvolucijska nevronska mreža vir:wikipedia

V delu mreže, kjer poteka ekstrakcija značilk, mreža preko digitalnih filtrov ekstrahira različne značilnosti iz slike. V primeru slike mačke bi bil to filter, ki bi zaznal smrček, dlako ali uhelj. Naslednji filter bi zaznal obliko smrčka, vzorec dlake ali obliko uhlja. V zadnjem delu mreže poteče postopek klasifikacije, kjer značilke uporabimo za razlikovanje pasem mačk. V tem primeru v zadnjem delu namesto klasifikacije poteče regresija.

4.2.2 PointNet

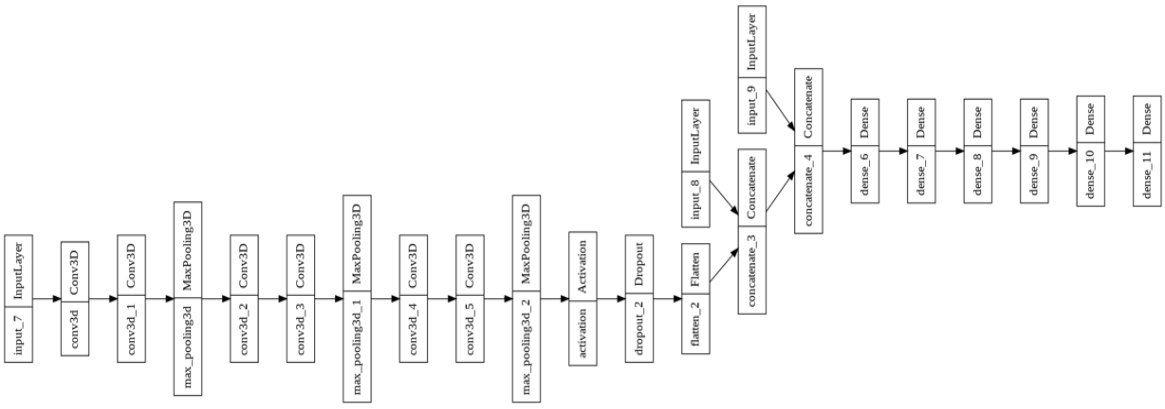
PointNet so razvili avtorji v članku [10]. Njihov cilj je bil razviti model, ki bi namesto vokslov uporabljal oblak točk kot vhodne podatke. Razvili so lahko (angl. light weight) mrežo, ki lahko konkurira trenutno najboljšim mrežam (angl. state of the art). Na spodnji sliki lahko vidimo arhitekturo PoinNet mreže, kot so si jo zamislili avtorji.



Slika 10: PointNet mreža

4.2.3 Mreža VOX

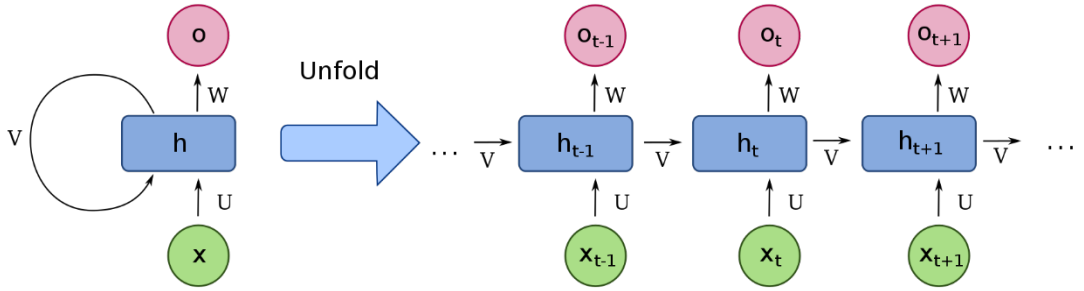
Mreža je sestavljena na osnovi članka [3]. Osnovna mreža je imela preveč povezav, da bi delovala na mojem sistemu, zato sem jo poenostavil. To sem dosegel z zmanjšanjem števila filtrov iz 16 na 8 in zmanjšanjem velikosti filtrov iz 7 na 3.



Slika 11: Arhitektura mreže VOX

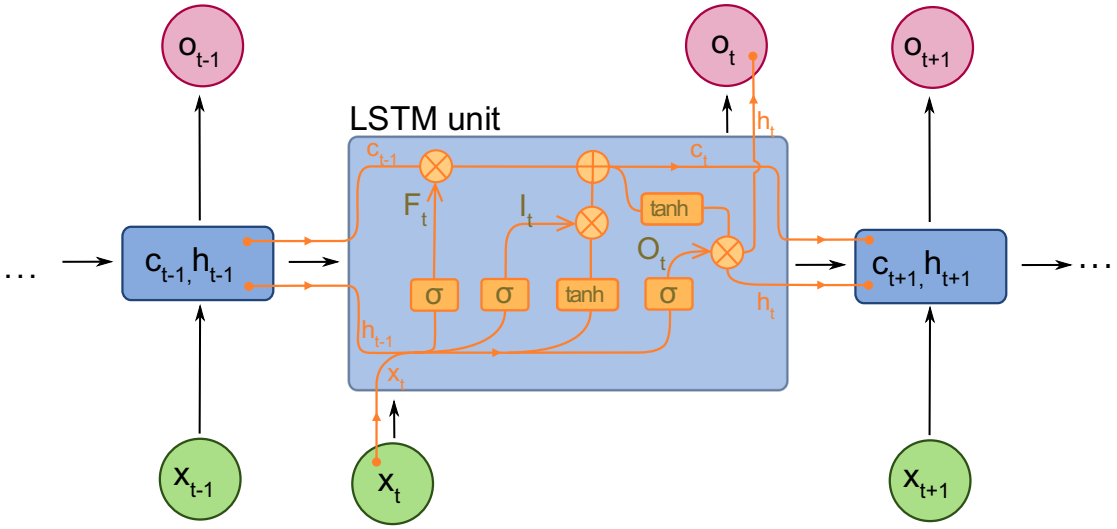
4.2.4 Long short term memory reccurent neural network (LSTM RNN)

RNN (angl. reccurent neural network) je nevronska mreža, ki pri izračunu gradienta upošteva vrednost prejšnjega stanja. RNN lahko uporabljajo svoje notranje stanje oziroma spomin za procesiranje časovno odvisnih podatkov, kot so govor, pisava in ostale časovne vrste.



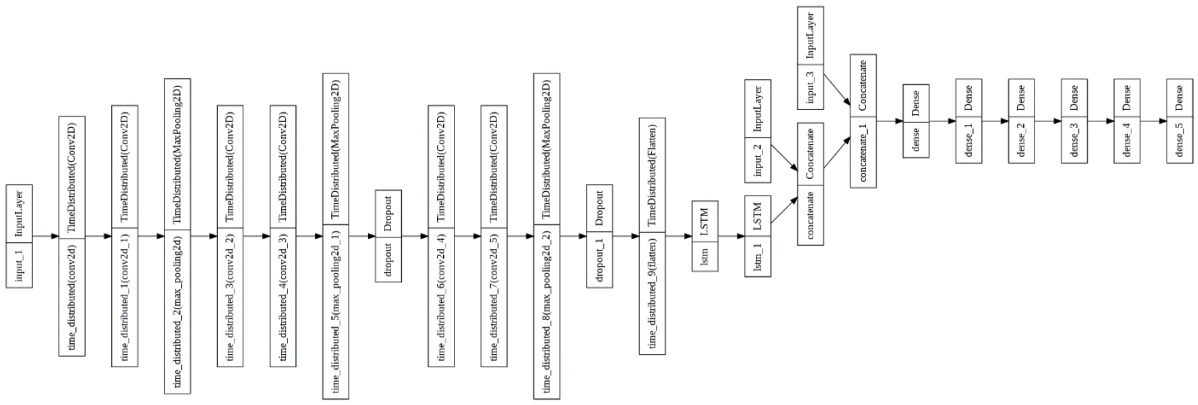
Slika 12: skica RNN vir:https://en.wikipedia.org/wiki/Recurrent_neural_network#Long_short-term_memory

RNN imajo številne probleme, kot recimo gradientne izgube, kjer se napaka skozi čas izgublja, kar pa poslabša rezultate optimizacije. Odgovor temu problemu je bila arhitekura LSTM (angl. Long short-term memory), ki je v jedru še vedno RNN, vendar za delovanje uporablja dodatne spominske celice oziroma vrata.



Slika 13: skica LSTM vir:https://en.wikipedia.org/wiki/Recurrent_neural_network#Long_short-term_memory

Čeprav je bil LSTM načrtovan za časovno odvisnost pri iskanju vzorcev, lahko izvedem 2D kovolucijo nad vsako izmed 100 slik in izhod napeljem v LSTM, ki bo iskal vzorce med posameznimi sloji modela.



Slika 14: Arhitektura mreže LSTM

4.2.5 Funkcije napake

Pri učenju sem za optimzacijsko funkcijo uporabil MAE, saj je bila ta za mojo aplikacijo bolj stabilna. Za analizo rezultatov bom uporabljal vse tri funkcije. Vse eksperimente bom ponovil 3 krat, kjer bom vsakič uporabil druge nakjučne vzorce. (angl. Shuffle and split).

4.2.5.1 Povprečna absolutna napaka (MAE)

Funkcija povprečne absolutne napake je definirana kot

$$\mathbf{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

$\mathbf{MAE}$ = mean absolute error
y_i = prediction
x_i = true value
n = total number of data points

MAE je absolutna povprečna razlika med X in Y. Značilnost te funkcije je, da je absolutna, torej se napake ne odštevajo.

4.2.5.2 Procentualna povprečna absolutna napaka (MAPE)

Procentualna poprečna absolutna napak je definirana kot

$$M = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

M = mean absolute percentage error
n = number of times the summation iteration happens
A_t = actual value
F_t = forecast value

MAPE nam poda podobno napako kot MAE, le da jo poda v odstotkih odstopanja od realne vrednosti. MAPE je zaradi njene intuitivnosti zelo pogosto uporabljena pri regresiji. Eden večjih problemov pri MAPE je deljenje z 0 oziroma ko je naša dejanska vrendost enaka 0.

4.2.5.3 Koren povprečja kvadrata napake (RMSE)

Koren povprečnega kvadrata napake je definiran kot

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}{N}}$$

RMSD = root-mean-square deviation
i = variable i
N = number of non-missing data points
x_i = actual observations time series
$\hat{x}_i$ = estimated time series

RMSE je v osnovi zelo podoben MAE, le da je zaradi kvadrata napake veliko bolj občutljiv na vzorce, ki bolj odstopajo od prave vrednosti oziroma imajo največji prispevek k seštevku. Tako bo optimizacijska funkcija najprej minimalizirala le-te.

4.3 Okolje in programi

4.3.1 Strojno učenje

Za strojno učenje sem uporabil Google Collab. To je spletna aplikacija, ki omogoča dostop do Googlovih računskih modulov (angl. compute modlues). Collab brezplačno ponuja tako CPU mašine kot tudi mašine z GPU pospeševalnikom za DLL operacije (angl. deep learning library). Modili pounjajo od 12GB do 16GB RAM-a.

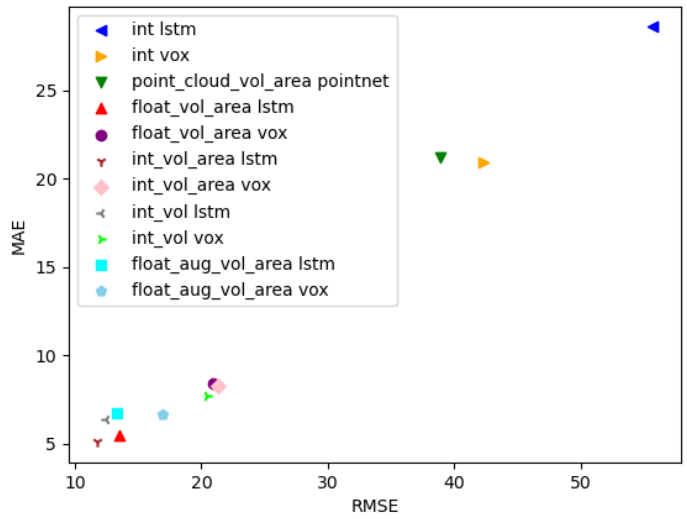
4.3.2 Programska oprema

Za sledenje razvoju in migracijo kode med mojim računalnikom in Collabom sem uporabljal GitHub. Večina kode je bila napisana v python 3.9. Za zapis matrik sem uporabljal numpy verzijo 1.19.5 [11], za strojno učenje pa tensorflow verzijo 2.7.0 [11].

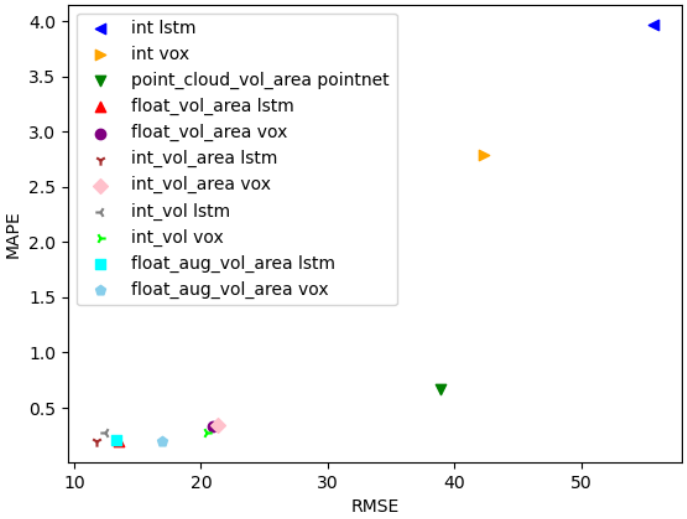
5 Rezultati

5.1 Graf vseh eksperimentov

Na grafu 1 in 2 vidimo vseh 11 izvedenih eksperimentov. Ime v legendi je sestavljeno iz dveh oziroma treh delov. Prvi nam pove tip zapisa podatkov (integer, float ali point cloud). Dodatek nam pove, ali je bila podana informacija o volumnu (vol) oziroma površini (area) modela ali pa če so bili podatki augmentirani (aug). Zadnja zveza nam pove tip mreže, ki sem jo uporabil (VOX, LSTM ali PointNet).



Graf 1: MAE in RMSE



Graf 2: MAPE in RMSE

Najboljše rezultate sem dobil z mrežo »LSTM INT«, s celoštevilskim tipom zapisa, volumnom in površino. Zelo blizu ji sledijo:

- LSTM mreža z decimalnim zapisom, volumnom in površino
- LSTM mreža z augmentiranimi decimalnimi podatki
- in LSTM mreža s celoštevilskim tipom zapisa, z volumnom brez površine.

Najslabše rezultate sem dobil s Point cloud, z volumnom in površino. Point cloud brez volumna in površine je imel tako slabe rezultate, da ga ni bilo mogoče uvrstiti na graf.

V nadaljevanju bom na podlagi rezultatov ugotavljal, katera mreža je najprimernejša, kateri tip zapisa podatkov je bolj primeren in pa kateri dodatki najbolj izboljšajo rezultat in koliko.

5.2 Primerjava podatkovnega tipa

Point cloud zapisa podatkov je bil v povprečju 4 x slabši od celoštevilskega in decimalnega zapisa. Večja natančnost decimalnih števil ne prinaša nobene izboljšave, celo obratno: prinaša slabše rezultate.

Tabela 1: Primerjava podatkovnega tipa

TIP NAPAKE	INT (N=2)	FLOAT (N=2)	POINT (N=1)
RMSE	16,55	17,19	38,86
MAPE	0,266	0,2645	0,67
MAE	6,70	6,97	21,24

5.3 Primerjava vpliva volumna in površine

Iz zgornje tabele opazimo, da se napaka drastično zmanjša z dodatkom informacije o volumnu. Če prostornini dodamo še površino, se napaka zmanjša, a le minimalno. Pri tem moramo upoštevati, da dodajanje te značilke ni podaljšalo časa učenja ali pa dodatno obremenjevalo procesor, kot to naredijo ostale izboljšave.

Tabela 2: Primerjava vpliva volumna in površine

TIP NAPAKE	BREZ (N=2)	VOL (N=2)	VOL AREA (N=2)
RMSE	49,00	16,44	16,54
MAPE	3,37	0,28	0,26
MAE	24,78	7,03	6,7

5.4 Primerjava vpliva avgmentacije podatkov

Avgmentacija podatkov v povprečju pripomore k boljši oceni, kar je bilo pričakovano, saj je imel učni set skoraj 5 krat več podatkov. Pri tem je treba upoštevati, da je bilo računanje vseh rotacij zelo zamudno in da se je čas učenja podaljšal iz 5 minut na 30 minut.

Tabela 3: Primerjava vpliva avgmentacije

TIP NAPAKE	BREZ (N=2)	AUG (N=2)
RMSE	17,19	15,12
MAPE	0,26	0,19
MAE	6,97	6,73

5.5 Primerjava mrež

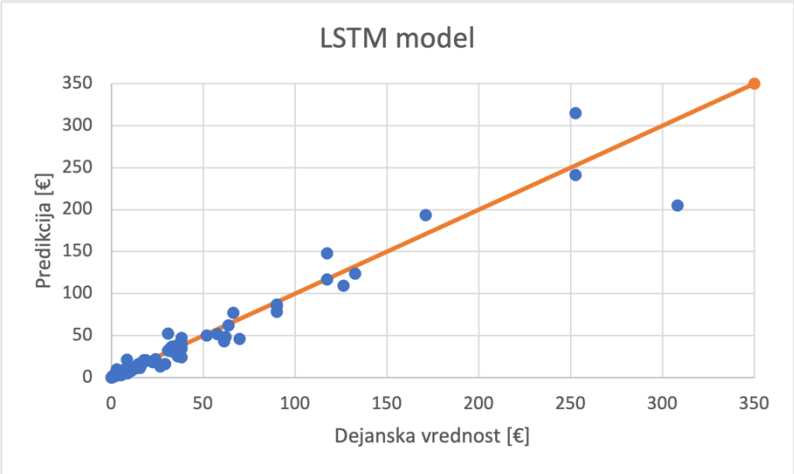
Podobno kot pri primerjavi zapisa podatkov se je posledično slabo odnesla tudi mreža pointnet. Razlog za slabo delovanje je lahko tako zapis podatkov kot tudi mreža. Zaradi nezadnostno velikega števila eksperimentov ni mogoče empirično dokazati izvora problema.

V primerjavi sta LSTM in VOX oba zelo dobra, vendar je LSTM še vedno za 6 % boljši od VOX-a.

Tabela 4: Primerjava mrež

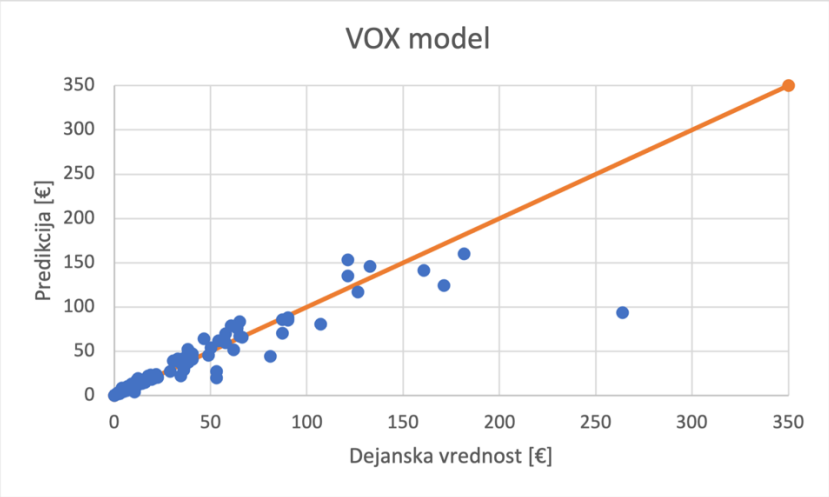
TIP NAPAKE	LSTM (N=4)	VOX (N=4)	POINTNET (N=1)
RMSE	12,75	19,90	38,86
MAPE	0,22	0,28	0,67
MAE	5,93	7,78	21,24

5.6 Korelacijska grafa za LSTM in VOX model



Graf 3: Korelacija med dejanskimi in predvidenimi vrednostmi za LSTM model

Razpršeni graf LSTM modela nakazuje, da predikcija deluje dobro. V območju nižjih cen model deluje boljše, v območju z višjimi pa malo slabše. Razlog za to je med drugim tudi to, da je v učnem setu velik del datotek nižje cenovnih. Podobne, a malo slabše rezultate lahko vidimo na spodjem grafu za VOX model.



Graf 4: Korelacija med dejanskimi in predvidenimi vrednostmi za VOX model

6 Razprava

Zgoraj predstavljeni rezultati nakazujejo, da so LSTM mreže primerene za to aplikacijo. Izkaže se, da so celo boljše od obstoječega pristopa s konvolucijskimi nevronskimi mrežami. Primer take mreže je bil VOX. Med pregledom dosedanjega dela nisem opazil, da bi se LSTM mreže aplicirale na tak način.

Cilj seminarske naloge je bil izdelati mrežo, ki zna oceniti ceno 3D tiska objekta in oceniti, katere lastnosti najbolj vplivajo na performanco mreže. Podatki so pokazali, da je zelo pomembna informacija o volumnu. Modeli brez te informacije so bili skoraj ne uporabni. Iz rezultatov je razvidno, da tip zapisa podatkov (int ali float) ne vpliva na rezultate. Razlog za to je, da sem uporabljene podatke samo pretvoril iz celoštevilskega zapisa v decimalni zapis, torej se je 1 zapisala kot 1.0 to pa ne dodaja neke dodatne informacije. Prednost decimalnega tipa zapisa se pokaže pri avgmentaciji podatkov, kjer objekt rotiramo in dobimo veliko večjo resolucijo pri decimalnem zapisu. Avgmentacija podatkov opazno vpliva na rezultate, predvsem MAPE.

V seminarski nalogi sem se zgledoval po članku [3] in se odločil za uporabo dimenzij 32^3 vokslov. V mojih prvih eksperimentih so te dimenzije nakazovale slabe rezultate. Razloga za to sta lahko dva. Prvi razlog je, da so avtorji v omenjenem članku uporabili preproste objekte, kot so vijaki, zobniki, krogle, valji itd. To so objekti oziroma modeli, ki se pogosto izdelujejo s CNC obdelavo. Takšne objekte lahko predstavimo z manjšim številom vokslov, brez da bi izgubili pomembne značilke, ki determinirajo ceno. V primerjavi s CNC obdelavo lahko s tiskanjem izdelamo objekte, ki so zelo kompleksi in vsebujejo veliko značilk. To pomeni, da za estimacijo cene 3D tiskanaja objektov potrebujemo večjo resolucijo objekta kot v primerjavi s CNC obdelavo. Drugi razlog bi lahko bil preprosto premajhno število vzorcev. Med tem, ko so avtorji v omenjenem članku uporabili 1000 vzorcev, sem jih jaz uporabil le 200. Posledično sem se odločil za uporabo dimenzij 100^3 vokslov, kar je rezultiralo boljše rezultate.

Nazadnje bi izpostavil težavo o količini in zaupnosti podatkov. Med tem, ko večje družbe ne bi imele težav pridobiti velikega števila vzorcev (1000+) za učenje nevrosnke mreže, bi tu nastal problem za mlade proizvajalce, ki imajo omejeno število vzorcev (50-100). Težava nastane tudi, ker proizvajalec podpiše NDA pogodbo o zaupnosti, sploh če gre za izdelavo prototipov. Rešitev tega problema je prenos znanja (angl. trasnfer learning). V poglavju 4.2.1 sem omenil, da je nevronska mreža zgrajena iz dveg delov. Prvi del je ekstrakcija značilk, drugi pa klasifikacija oziroma regresija. Če uporabimo že prej naučeno mrežo, kot je VGG [13] ali

RESNT [14] lahko zaklenemo uteži v prvem delu in spreminjamo uteži le v drugem. Za naše potrebe bi lahko s pomočjo prosto dostopnih podatkov naučili splošno mrežo, potem pa za potrebe posameznega proizvajalca spremenili uteži v drugem delu mreže. Ta pristop bi potreboval veliko manj podatkov kot klasičen.

7 Zakjuček

Modela LSTM in VOX se lahko primerjata z najboljšimi mrežami, ki jih je zasnovala skupnost ML. MAPE, ki sem ga dosegel, je bil 22 %. Za praktično uporabo te metode v industriji bi se morala ta napaka zmanjšati vsaj pod 5 % . Ta napaka je z omenjeno metodo dosegljiva, če se uporabi dovolj velik podatkovni set. Znano je, da perfomranca nevronske mreže ob večanju števila podatkov narašča zelo dolgo, preden perfomančna krivulja doseže plato. Poleg večanja števila podatkov bi z močnejšo strojno opremo lahko uporabili globlje mreže, katere bi prepoznale še več značilk. Implementacija mreže trenutno gleda odvisnost le v eni smeri, oziroma od spodnje plastnice do zgornje. Dodatno izboljšavo bi lahko doprinesla implementacija, kjer bi mreža spremljala odvisnost v obe smeri (angl. Bidirectional LSTM) po zgledu avtorjev [15]. Prav tako bi se izboljšave lahko naredile na strani pretvarjanja podatkov iz STL v voksle, kjer bi namesto trenutnega preporsega algorima uporabili bolj točne postopke, kot jih recimo opisujejo avtorji v [16]. Izboljšave bi lahko naredili praktično na vsakem koraku, kar bi nam omogočilo doseči mejo 5 % in posledično praktično uporabo te metode v industriji.

8 Literatura

[1] A. Mahadik, *Cost Estimation of Layer Additive Manufacturing Using Break-down Approach,* Ohio University, 2018.

[2] „FAMA3D,“ 2021. [Elektronski]. Available: https://www.fama3d.it/en/preventivo.html. [Poskus dostopa 8 1 2022].

[3] N. K. Soyoung Yoo, „Explainable Artificial Intelligence for Manufacturing Cost Estimation and Machining Feature Visualization,“ *ArXiv,* št. v1, 2020.

[4] Y. L. Y. W. Siu L. Chan, „Data-driven cost estimation for additive manufacturing in cybermanufacturing,“ *Journal of Manufacturing Systems ,* Izv. 46, pp. 155-126, 2018.

[5] MakerBot, „Thingsverse,“ MakerBot, 2022. [Elektronski]. Available: https://www.thingiverse.com. [Poskus dostopa 13 1 2022].

[6] Ultimaker, *CuraEngine,* https://github.com/Ultimaker/CuraEngine, 2017.

[7] M. Dawson-Haggerty, *Trimesh [Computer software],* https://github.com/mikedh/trimesh, 2019.

[8] C. Schranz, *"Tweaker - Auto Rotation Module for FDM 3D Printing,* Salzburg: https://github.com/ChristophSchranz/Tweaker-3, 2016.

[9] C. Pederkoff, *stl-to-voxel,* https://github.com/cpederkoff/stl-to-voxel, 2020.

[10] H. S. K. M. L. J. G. Charles R. Qi, „PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation,“ v *Computer Vision and Pattern Recognition (VCPR)*, 2017.

[11] A. A. P. B. Martín Abadi, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems,* 2015.

[12] M. K. v. d. W. S. Harris CR, „Array programming with NumPy,“ *Nature,* Izv. 585, pp. 357-362, 2020.

[13] A. Z. Karen Simonyan, *Very Deep Convolutional Networks for Large-Scale Image Recognition,* arXiv, 2014.

[14] X. Z. S. R. J. S. Kaiming He, *Deep Residual Learning for Image Recognition,* ariXiv, 2015.

[15] M. S. a. K. K. Paliwal, „Bidirectional recurrent neural networks,“ *IEEE Transactions on Signal Processing,* Izv. 47, št. 11, pp. 2673-26811, 1997.

[16] R. Y. V. F. a. Y. K. Jian Huang, *An Accurate Method for Voxelizing Polygon Meshes.*

Vsa uporabljena koda je na voljo na https://github.com/jenkoj/mppe