Zimný semester:

Ročníkový projekt sa sústreďuje na zber datasetov pre trénovanie neurónových sietí.

Zber pozostáva z 4 častí:

1. Zber kódu
2. Zber skompilovaného kódu
3. Zber obrázkov
4. Zber textu

Zber kódu

V tejto časti som sa rozhodol využiť API githubu. Najskôr som skúšal pomocou knižnice requests robiť requesty priamo na github api neskôr som prešiel na knižnicu PyGithub ktorá má v sebe implementované základne volania tohoto api. Týmto spôsobom som schopný získať reálny kód v konkrétnych jazykoch. Script teda cez github api zistí aký jazyk sa v repozitári používa. Potom repozitár stiahne a vytiahne z neho všetky súbory ktoré používajú suffix tohoto jazyka (jazyk + sufix pár získa z konfiguráku)

Vysvetlenie config.json file-u:

github_access_key: tento kľúč možno získať z githubu. Slúži na pripojenie do github api
max_repos: maximálny počet repozitárov na preverenie. Ak by sme preverili viac ako tento počet repozitárov script sa zastaví.
languages_with_suffix: Sem zapisujeme všetky jazyky ktoré chcem vyhľadať (mená treba použiť z github api). Zapisujeme ich ako pár “meno”: “suffix súborov v tomto jazyku” .
output_dir: miesto kde sa budú ukladať dáta.
required_bytes: Počet bytov ktoré sa script pokúsi získať z každého jazyka.

Zber skompilovaného kódu

Zber skompilovaného kódu bol problémový, keďže získať ľahko kompilovateľný kód na rôzne architektúry nebolo úplne jednoduché (repozitáre z githubu by bolo ťažké skopmilovať). Preto v tejto časti používam veľmi jednoduché c kódy zo stránky https://beginnersbook.com/2015/02/simple-c-programs/. Tieto kódy sa pomocou Beautiful soup stiahnu a uložia. Potom sa pomocou scriptov z konfiguráku skúsia skompilovať na rôzne architektúry. Ak pri kompilácií nastane nejaký problém súbor sa preskočí.

Vysvetlenie config.json file-u:
architectures: tu sa zapíšu požadované architektúry s ich skriptami na kompilovanie. Nejaké príklady sú uvedené v config.example.json. Vstupný a výstupný súbor sú nahradené za INPUT_PATH a OUTPUT_PATH. Tieto miesto potom script nahradí konkrétnymi menami súborov.
output_dir: Miesto kde sa budú dáta ukladať.

Zber obrázkov

Na zber obrázkov som použil google search api. Toto api mi dovoľuje získať url obrázkov v požadovaných formátoch. Tie potom cez knižnicu requests ukladám. Na vyhľadávanie som použil googlesearchapiclient knižnicu keďže cez ňu to bolo jednoduchšie na implementáciu. Na získavanie týchto dát si ale v googli treba získať api key a google_cse_id. Tie možno získať zadarmo (ak sa ale prekročí 100 requestov za deň google ho na ten deň zablokuje).

Vysvetlenie config.json file-u:
google_api_key: api kľúč získaný z googlu.
google_cse_id: cse id získane z googlu.
queries: queries použité pri vyhľadávaní obrázkov. Každá query sa použije na maximálne 10 obrázkov (aby sa zabránilo tomu že neurónová sieť sa naučí hľadať niečo konkrétne na obrázku)
formats: Požadované formáty.
output_dir: Miesto kde sa budú dáta ukladať.
required_bytes: Počet bytov ktoré sa script pokúsi získať z každého formátu obrázku.

Zber textu

Na zber textu som najskôr uvažoval nad sociálnymi sieťami potom sa ale tento spôsob pre množstvo slangu a gramatickej neprestnosti zavrhol. Miesto toho som použil Wikipédiu. Na začiatku som sa snažil získať dáta sám cez BeautifulSoup (podobne ako pri kompilovaní) potom som ale objavil knižnicu Wikipedia kde je už BeautifulSoup pre wikipédiu implementovaný. Script stiahne z každého požadovaného jazyka obsah stránok ktoré sa na wikipédií pod search queries (výrazy ktoré sa zadajú do wikipédie aby nám navrhla existujúce články) nachádzajú. Ďalej sa tieto stránky skúsia uložiť aj v požadovaných formátoch (Ak sa nejaký obsah nepodarí uložiť v danom formáte stránka sa pre tento formát preskočí).

Vysvetlenie config.json file-u:
langs: Požadované jazyky.
queries: Queries na wikipédiu.
formats: Požadované formáty (v ktorých vie python encodovať).
output_dir: Miesto kde sa budú dáta ukladať.

Príklady na tieto configuráky sa vždy nachádzajú v config.example.json súboroch.

Pre rozbehanie projektu je v požadovanie directory nutné skopírovať tento príklad do config.json a potom zadať konkrétne hodnoty.

Ak chcete skúsiť len jednu konkrétnu časť každá Časť obsahuje standalone script ktorý pustí danú podčasť bez ostatných.

Konvertovanie na .bin súbory

Posledná časť tejto práce je prekonvertovať dané súbory na bin súbory ktoré vieme už použiť pri cvičení neurónových sietí. To prebieha nasledovne: Zavolajú sa všetky vyššie uvedené scripty. Potom sa všetky súbory toho istého typu spoja do jedného bin súbora a vymaže sa im prvých a posledných N byt-eov (toto by malo zabrániť sieťam učiť sa podľa hlavičiek). Nakoniec sa tieto súbory uložia do stage2_output_dir z konfiguráku.

Vysvetlenie config.json file-u:

stage1_output_dir: tu sa uložia dáta z prvej Časti (zber dát).
stage2_output_dir: tu sa uložia dáta z druhej Časti (spracovanie na bin súbory).
remove_bytes_start: počet prvých bytov na zahodenie.
remove_bytes_end: počet posledných bytov na zahodenie.
code_config: config z kód scriptu (viď vyššie).
compiled_code_config: config z kompilovaných kódov(viď vyššie).
images_config: config z obrázkov (viď vyššie).
text_config: config z zberu textu (viď vyššie).

Letný semester

Priebeh práce

Pre tento semester som sa po diskusii s kolegom Martinom Pašenom (po ktorom so projekt prebral) rozhodol experimentovať s rekurentnou neurónovou sieťou. Martin mi pripravil repozitár s jednoduchou RNN ktorá sa cvičila na náhodných dátach.

Najprv som implementoval dataset aby používal miesto náhodných dát, dáta z binárnych súborov (ktoré som získal z práce z minulého semestra).

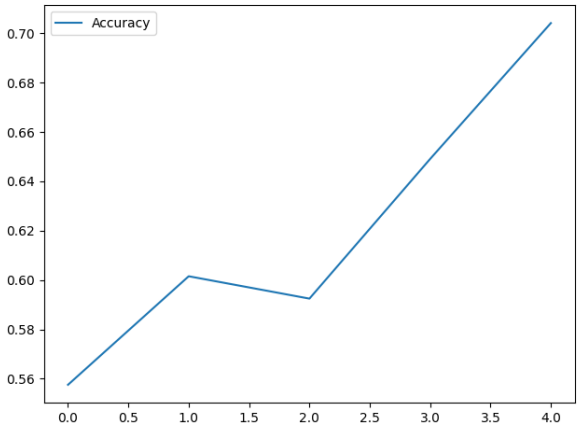
Potom sa mi podarilo zažiť to, že niekedy mi hodnoty ktoré mi vracala neurónová sieť prestrelili veľkosť maximálnej hodnoty floatu a nevedel som prečo. Po bližšom preskúmaní a identifikovaní tohto problému sa mi podarilo zistiť že sa jednalo o takzvaný gradient explotion ktorý sa stáva neurónovým sieťam. Nakoniec sa mi podarilo upraviť sieť tak aby sa pred ním dokázala brániť.

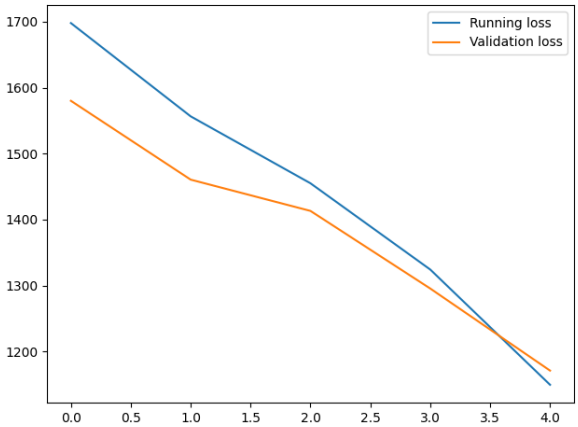
Ďalej som prepracoval neurónovú sieť aby dokázala pracovať na grafickej karte (pre zrýchlenie učenia) a upravil vstupy, aby sa grupovali do takzvaných batchov (Znamená, že neurónová sieť trénuje s x dátami naraz miesto po jednom). Tento prístup veľmi urýchlil možnosť cvičenia tejto siete.

Ďalší krok bolo rozdelenie datasetu na takzvaný training a validation dataset. Pri tejto práci som zároveň implementoval možnosť vykreslovaňia accuracy a validation loss na grafoch aby sme mali možnosť lepšie sledovať ako sa neurónovej sieti darí.

Po experimentovaní s dátami a sledovaní výsledkov sme s kolegom Pašenom došli ku záveru, že by sme mohli skúsiť upraviť vstupy tak aby miesto čísla v intervale 0-255, posielali pole dĺžky 256 obsahujúci hodnoty true a false. Tento prístup by mohol zlepšiť presnosť za cenu dlhšej ceny cvičenia siete.

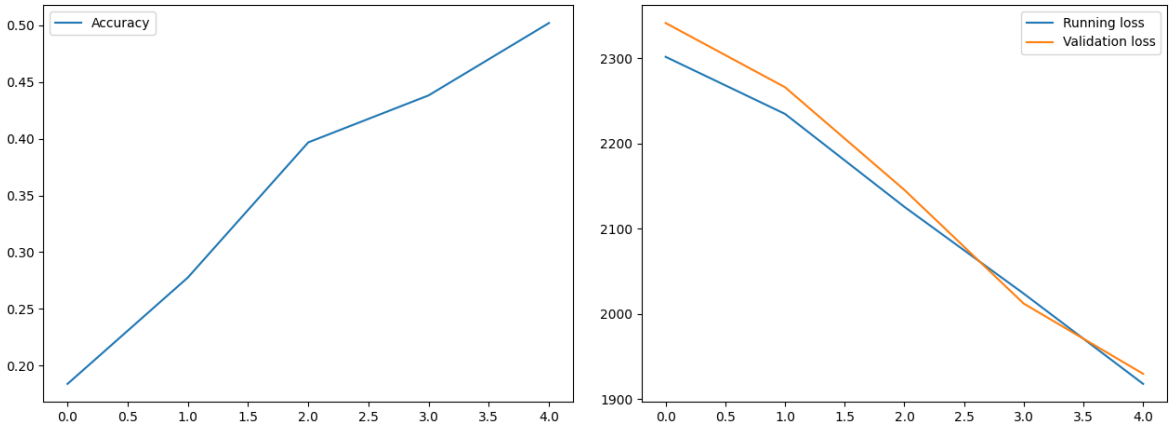
Výsledky cvičenia prvej siete (validation loss je loss testovacieho datasetu a running loss je loss na cvičiacom datasete, accuracy je počítaná na testovacom datasete):



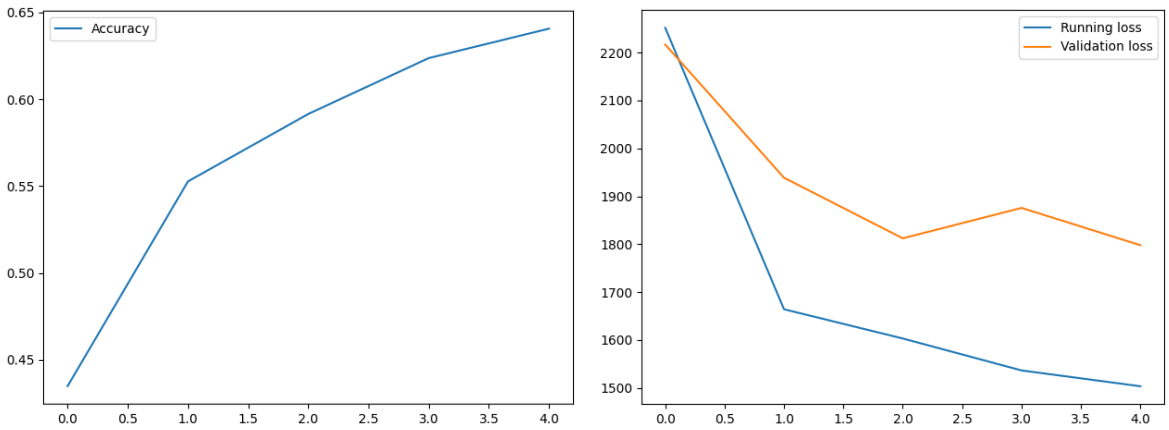


Zaujímavý jav ktorý tu môžeme sledovať je to, že validation dataset mal prvé 4 epochy nižší loss ako dataset, na ktorom sa priamo cvičilo (pre informáciu tento loss nie je ten istý ktorý som používal pri cvičení, ale po cvičení som neurónovú sieť pustil na datasete ešte raz aby som získal priemerný loss).

Dataset použitý vyššie bol ale zjednodušený dataset (ktorý neobsahoval šum, jpg obrázky a ďalšie zložitejšie formáty.), a preto som sa rozhodol pustiť túto neurónovú sieť ešte na zložitejšom datasete (obsahujúcom viac typov obrázkov, kódy v rôznych jazykoch, kompilované kódy pre viac druhov zariadení, a pre text v utf-8 a utf-16 (vo viacerých jazykoch)).



Ako môžeme vidieť tu sme po 5 epochách dosiahli presnosť len asi 50 percent a loss je tiež vyšší. Z tohto dôvodu som skúsil vytvoriť lepší spôsob vstupu, ktorý by aj na tomto datasete dosahoval lepšie výsledky. Na vstup som teda posielal pole dĺžky 256 ktoré na príslušnom indexe dosahovalo 1 (podľa hodnoty bajtu na vstupe) a na ostatných miestach bola nadstavená 0 (použil som bool)



Ako vidíme tu sa nám podarilo dostať ku lepšej accuracy a loss. Bohužiaľ pre veľkosť vstupu postup uvedený vyššie bol časovo náročnejší keďže vstup v takomto formáte nevošiel naraz do pamäte grafickej karty. Aby som teda vôbec mohol takéto cvičenie pustiť vyriešil som to tak, že vstup bol uchovávaný v pôvodnom formáte a do tohto formátu sa transformoval až tesne pred použitím.

Vidíme teda že tieto siete dokážu dosiahnuť istú mieru úspešnosti a myslím si, že táto miera sa dá ďalej zdokonaľovať. Pre budúce pokusy navrhujem vylepšiť dataset tak, aby neurónová sieť dokázala získavať vstupy rýchlejšie. Toto by možno bolo dosiahnuteľné tak, že by sme na vstup posielali polia dĺžky 8 (v klasickom formáte bitov). Ako som už poukázal na grafoch vyššie presnosť sa rapídne znížila pridaním zložitejšieho datasetu. Pre ďalší výskum by som teda taktiež navrhol lokalizáciu toho, ktoré vstupy robia sieti najväčší problém. Možnosť bežania výpočtov na výkonnejšom počítači by pravdepodobne taktiež urýchlila prácu a pomohla mi sledovať väčší počet epoch.