\hspace{10mm}Pri návrhu nášho riešenia sa zameriame na metódy učenia s učiteľom pomocou hlbokých neurónových sietí. Pričom budeme používať konvolučné neurnové siete, ktoré najlepšie dokážu rozoznávať vzory v obrazových dátach. Našim hlavným cieľom bude porovnať viaceré prístupy tvorenia filtrov v prvých vrstvách neurnových sieti. Prístupy na skúmanie a porovnávanie sme si vybrali backpropagation, autoencoder, transfer learning a filtre generovane pomocou Gáborovej funkcie, teda Gáborové filtre. Jednotlivé princípy dokážeme rozdeliť do dvoch kategórii a to podlá toho ako tvoria filtre teda automaticky, ktoré sieť sama inicializuje a manuálne mi dodáme sieti filtre, ktoré už ďalej upravovať nebude. Hlavnou myšlienkou porovnania bude pripojiť spomínané prístupy tvorenia filtrov do existujúcej architektúry na začiatok a následne trénovať na datasetoch, ktoré máme prístupne. Ako existujúcu architektúru použijeme Unet neurónovú sieť s hĺbkou 9, ktorá sa skladá zo špeciálnych konvolučných vrstiev, ktoré obsahujú 2 konvolučných vrstvy, návrh tejto architektúry môžeme vidieť na obrázku č. \ref{fig:architektura}. Túto architektúru sme vybrali kvôli jej najlepším výsledkom, pričom v prvotných testoch dosiahla veľmi dobrú presnosť a to 80\%. Iné architektúry, ktoré sme skúšali, nedosahovali tak dobré výsledky, pričom skúšané boli ResNet s hĺbkou 8, VGG16 a vlastne neurnové siete s maximálnou hĺbkou 5 a minimálnou 2.

\hspace{10mm}Cieľom jednotlivých prístupov a celkovo neurónovej siete bude dosiahnuť čo najlepšie výsledky, ktoré budeme merať vo viacerých metrikách ako presnosť (accuracy), precíznosť (precision) tiež nazývaná ako pozitívna prediktívna hodnota, (recall) tiež nazývaná ako citlivosť a f1 skóre, ktoré je kombináciou precíznosti a recall. Jednotlivé prístupy budeme modifikovať a zisťovať, ktoré kombinácie parametrov vykazujú najlepšie výsledky, Parametre, ktoré budeme meniť v prístupoch budú záležať na prístupe, ale hlavne chceme meniť veľkosť daných filtrov (kernelov) a počet týchto filtrov. Ostatne vrstvy, teda iné ako tie v jednotlivých prístupoch nebudeme meniť už počas trénovania a porovnávania prístupov. Úlohu, ktorú budú mať prístupy je jednotná a to klasifikovať do dvoch tried, čo sa týka histologických dát \ref{text:pcam}, ktoré použijeme budeme klasifikovať na malígne a nemalígne. A dáta štrukturovaných povrchov \ref{text:dagm} klasifikovať na obsahujúci objekt a neobsahujúci objekt.

\begin{figure}[h!]

\begin{centering}

\includegraphics[width=15cm]{assets/images/300\_1\_1.png}

\par\end{centering}

\caption{Návrh architektúry Unet a s jednotlivými prístupmi. \label{fig:architektura}}

\end{figure}

\subsection\*{Prístup backpropagation}

\hspace{10mm}Prvý prístup, ktorý budeme používať a testovať je backpropagation, zaradiť tento prístup môžem do skupiny automatickej inicializácie filtrov. Tento prístup sme vybrali kvôli tomu, že nie je nutná nejaká špeciálna implementácia a teda je to najviac používaný prístup, v doméne konvolučných neurónových sieti. Taktiež tento prístup je veľmi efektívny a dáva sieti skoro úplnú kontrolu o tom ako bude extrahovať vzory a ucíť sa ich. Čo sa týka nášho zadania, aby sme dodŕžali architektúru Unet neurónovej siete, použijeme presne takú istú vrstvu pre tento prístup ako v celej architektúre, A teda bude zložená z dvoch konvolučných vrstiev, pričom jedna bude skrytá, batchnormalization vrstvy, aktivačnej funkcie ReLu, združovacej (pooling) vrstvy a dropout vrstvy pre predídeniu pretrénovania. Pre tento prístup sme zvolili ako vhodné kombinácie, veľkosť filtrov 3x3, 5x5 a 7x7 a počet filtrov 16, 32 a 64.

\subsection\*{Prístup autoenkóder}

\hspace{10mm}Ako druhý prístup, ktorý budeme používať a testovať je autoencoder. Tento prístup je opísaný v časti \ref{autoenkoder}. Keďže tento prístup, metóda, ma za úlohu skopírovať a následne prekopírovať dáta, potrebuje viac ako jednu konvolučnú vrstvu, a pre efektívnejšie vykonanie tejto metódy, použijeme až 4 konvolučné vrstvy, pričom budeme spájať prvú s poslednou pomocou zlučovacej (Concatenate) vrstvy a pre-vzorkovacou (UpSampling) vrstvou. Dôvod výberu tohto prístupu je ten, že si myslime, že pri jeho vykonávaní, teda skopírovaní a následnom prekopírovaní, by sa mohol naučiť veľmi doležíte informácie, vzory o dátach, ktoré automaticky aplikuje na dáta, a svoj výstup pošle ďalej. Teda dáta, ktoré budú obsahovať pozitívnu klasifikáciu by po výstupe z tejto vrstvy mali vyzerať podobne čo pomôže ďalej v klasifikácii. Tento prístup sa radi taktiež do triedy automatickej inicializácie filtrov. Pre tento prístup sme zvolili ako vhodné kombinácie, veľkosť filtrov 3x3, 5x5 a 7x7 a počet filtrov 16, 32 a 64.

\subsection\*{Prístup Transfer learning}

\hspace{10mm}Ďaľší prístup, ktorý použijeme a otestujeme je transfer learning. Opis tohto prístupu je opísaný v časti \ref{transferlearning}. Ako prevzatý model, ktorý použijeme budú už dobre známe modely, ktoré sú voľne dostupne aj s naučenými váhami, teda nebudeme trénovať pre tieto účely vlastné modely. Ako vhodné kombinácie pre tento prístup, nie sú podobné varianty ako u predošlých dvoch prístupoch, veľkosti filtrov a počet filtrov, keďže tieto kombinácie a parametre sú presne definovane v modeloch, ktoré použijeme. Jedinou vhodnou kombináciou bude počet filtrov, ktorým umožnime znova sa ucíť, a typ modelu, teda model, ktorý použijeme. My sme sa rozhodli použiť modely, ktoré majú veľmi dobre úspešnosti čo sa týka klasifikácie v obrazových dátach, teda použijeme model ResNet152 a VGG16. Dôvod výberu tohto prístupu plynie z prác, ktoré sme opisovali v časti \ref{publikacie}, pričom veľká cýst novodobých riešení je založená na tomto princípe. Preto veríme, že tento prístup by mohol mat veľmi dobré výsledky vďaka povahe, fungovania, tohto prístupu. Očakávame, ale aj väčší nárast trénovacieho a predikčného času pre veľkosť výslednej architektúry po pripojení modelu k nasej architektúre (Unet). Tento prístup môžem zaradiť znova do skupiny automatická inicializácia filtrov, čí už kvôli tomu, že filtre už boli vytvorené podobných stolom, alebo tomu, že určitú cýst filtrov, ktoré použijeme v modely znova preučíme.

\subsection\*{Prístup Gáborové filtre}

\hspace{10mm}Posledný prístup, ktorý chceme použiť a otestovať je vytvorenie filtrov pomocou Gáborovej funkcie, pričom tie vložíme ako jediné filtre, ktoré tato vrstva bude mať a nebude môcť ich meniť. Tento prístup sme vybrali kvôli tomu, že veríme, že Gáborové filtre budú mať dobré výsledky a budú vedieť lepšie nájsť a zdetegovať hrany poprípade objekty v dátach, keďže sú generovane prirodzenou funkciou a vyjadrujú určitú prirodzenosť používanú v prírode, ako spomíname aj v odstavci \ref{gaborfilter}, kde taktiež opisujeme ako sa tvoria a ako vyzerá Gáborová funkcia. V tomto prístupe pre lepšie použitie budeme používať väčšie filtre ako v iných prístupoch, pričom počet filtrov necháme rovnaký ako v iných kombináciách jednotlivých prístupoch. Použijeme veľkosť filtrov 5x5, 7x7 a 9x9. Kvôli skutočnosti, že tieto filtre sa budú musieť tvoriť manuálne, očakávame, že aj používanie bude oveľa zdĺhavejšie a teda cash, počas ktorého sa bude trénovať sieť bude rovnako ako v prístupe transfer learning oveľa väčší. Tento prístup sa zaraduje do skupiny manuálnej inicializácie filtrov. Tieto filtre budeme pri každom modelu vytvárať nanovo a to pomocou dostupných funkcii. Očakávame, že implementácia tohto prístupu bude najkomplikovanejšia v porovnaní s ostatnými. Pričom budeme musieť ešte otestovať vhodne vybratie parametrov pre Gáborovú funkciu. To sa budeme snažiť tak aby rôznorodosť týchto filtrov bola čo najväčšia.

\subsubsection{PCAM dataset (\ref{text:pcam})}

Načítanie a používanie datasetu PCAM nebolo veľmi náročne, pretože spolu s dátami bol poskytnutý aj skript pre načítanie tohto datasetu. Celý datasete je rozdelený na trénovaciu, validačnú a testovaciu časť, každá časť bola rozdelená do viacerých súborov. Xové dáta, teda obrázky sú uložene do HDF5 matice, čo uľahčuje použitie, šetri pamäťové miesto a celkový čas potrebný pre načítanie týchto dát. Čo sa týka yových dát tie sú uložene taktiež v HDF5 súbore, pričom jediná potrebná úprava bola zmeniť tvar, to sme vykonali pomocou knižnice Numpy. Každá časť obsahuje tiež aj meta súbor, ktorý je vo formáte csv. V tomto súbore sa nachádzajú x a y koordináty, či sa tumor na týchto dátach nachádzate, či je obrázok centrom tumoru, a WSI.

\subsubsection{DAGM dataset (\ref{text:dagm})}

Práca s týmto datasetom bola náročnejšia než s predošlým, pretože v tomto datasete boli dáta rozdelené na holé obrázky, ďalej o anotované obrázky, kde boli len obrázky s pozitívnou triédrov a tieto obrázky boli vyznačene bielou farbou pričom ostatné pixeli, kde sa pozitívna trieda nenachádzala boli čierne. Spolu s tým bol poskytnutý txt súbor, v ktorom boli označené pozitívne a negatívne triedy s názvom obrázka. Jednotlivé triedy datasetu, ako som spomínal v časti \ref{text:dagm}, boli rozdelené do trénovanej a testovacej časti. Keďže tento dataset obsahoval obrázky vo veľkosti 512x512, a nás pôvodný dataset PCAM, obsahoval obrázky 96x96, museli sme upraviť obrázky, aby dosiahli veľkosť 96x96. Celý proces spracovania týchto obrázkov fungoval tak, že vytiahli sme len tie pozitívne obrázky a následne len tie sme orezávali.

Celé orezávanie bolo robene tak, že sme podlá nastavených parametrov sa posúvali po obrázku, v základe to bolo 16 pixelov doprava, pokiaľ sme neprišli na koniec riadku potom sme sa posunuli 16 pixelov dole a znova až pokiaľ sme neprišli na koniec obrázku. Túto operáciu sme robili aj pre obrázky, kde sa nachádzali len o anotované a obrázky s reálnymi dátami, pričom celý dataset sme ukladali do h5py matice, kde sme aj ukladali yovú hodnotu a teda triedu či je obrázok pozitívny alebo negatívny. Na obrázku č. \ref{fig:createDAGM} môžeme vidieť popísaný algoritmus aj s výrobou pre triedu 6.

Vďaka tomu, že si vytvorenie tohto datasetu zabezpečujeme samy, vedeli sme ho jednoducho načítať a používať. Jedinou úpravou bolo len následne rozdelenie testovacej časti na validačnú a testovaciu cýst. Spôsob akým sme načítavali tento datasete môžeme vidieť na obrázku č.

Jednou z najdôležitejších častí implementácie neurónových sietí, je model. My sme ako hlavnú časť nášho model vybral architektúru Unet, k nej budeme vkladať rôzne prístupy tvorenia filtrov, teda k existujúcej Neurónovej siete Unet pripojíme novú vrstvu, ktorá svoje výsledky ďalej pošle. Na obrázku č. \ref{impl:unet} môžeme vidieť implementáciu Unet architektúry. Kde jednotlivé konvolučné vrstvy sú definované nasledovne \ref{impl:conv}.

Ako sme už spomínali v časti \ref{narv:back}, prístup backpropagation, bude kvôli lepšiemu priblíženiu k architektúre a ku kvalitnejšiemu porovnaniu s ostatnými prístupmi, použijeme na tvorbu tohto prístupu tradičnú konvolučnú vrstvu pre Unet architektúru, tu moste vidieť reálnu implementáciu. \ref{impl:back}. Na ďalšom obrázku moste vidieť vizualizáciu niekoľkých filtrov pre tento prístup. \ref{impl:filter:back}

Ďalšou neobytnou implementáciou bol prístup autoencoder, spomínaný v časti \ref{narv:auto}. Tento prístup obsahuje 4 konvolučné vrstvy pričom ku koncu siete spájame prvú s poslednou pomocou spájacej vrstvy a nad vzorkovacej vrsvy. Výstupom je ako u všetkých vrstva, ktorú môžem ďalej použiť. Jej implementáciu môžeme vidieť na obrázku č. \ref{impl:auto} a vizualizáciu jej filtrov na obrázku č. \ref{impl:filter:auto}

Posledná implementácia špeciálnych prístupov nachádzajúca sa vo skripte adaptive\\_model.py je Gáborové filtre. Tu sme implementáciu museli rozdeliť do dvoch krokov, v prvom kroku, v ktorom definujeme obyčajnú konvolučnú vrstvu, ktorej zakážeme učenie a upravovanie filtrov, a krok, kde vytvárame jednotlivé filtre. Toto rozdelenie bolo neobytne pre úspešnú implementáciu tohto prístupu. Išlo o to, že keby chceme hneď vložiť vygenerovane filtre pomocou knižnice OpenCv, tak sme museli vytvoriť tvar týchto filtrov, ktorý bolo nesmierne náročne vytvoriť. Preto sme zistili, že jednoduchšie bude upraviť váhy pre určitú vrstvu. Pričom sme zistili, že sieť si vytvorí váhy až potom ako sa definuje modelu. Implementáciu prvého kroku môžeme vidieť na obrázku č. \ref{impl:gabor}. V druhom kroku sme si museli najprv získať daný tvar váh pre konvolučnú vrstvu a následne podlá tohto tvaru vytvoriť Gáborové filtre. Následne vytvoriť dostačujúci počet týchto filtrov. Popis tohto algoritmu môžeme vidieť na obrázku č. \ref{impl:gaborfiltre}. Vizualizáciu jednotlivých filtrov môžeme vidieť na obrázku č. \ref{impl:filter:gabor}.

Ostatné funkcie v skripte adaptive\\_model.py sú len pre sprehľadnenie finálneho kódu, ktorý bude opísaný v nasledujúcej časti. Tieto funkcie len vytvárajú model, funkcia define\\_model, definujú vstup, funkcia input\\_layer, definujú výstup, funkcia output\\_layer, a prechodný výstup z obrazka do finálnej vrstvy pre určite prístupy, teda pre transfer learning, je špeciálny výstup po sieti a to GlobalAveragePooling, funkcia transfer\\_layer a pre ostatné prístupy to je vrstva s Flatten, ktorá premieňa obrázok na vektor, funkcia flatten\\_layer.

\hspace{10mm}V prvej časti tohto notebooku, len importujeme knižnice, ktoré sú potrebné pre ďalšiu prácu. Následne umožňujeme, aby výpočty sa uskutočňovali na grafickej karte.

\hspace{10mm}Ďalšie časti ako Load DAGM data a Load PCAM data, slúžia len na načítanie datasetom a rozdelenie ich do premenných pre následne trénovanie, testovanie a validáciu.

\hspace{10mm}V časti Build model, si definujeme základne globálne premenné pre trénovanie, ako tvar vstupu (shape), počet epoch, teda počet iterácii, koľkokrát chceme model trénovať (epochs), počet predikovaných tried (num\\_classes), teda na pozitívne a negatívne. A veľkosť trénovanej jednotky (batch\\_size).

\hspace{10mm}V ďalších častiach definujeme jednotlivé prístupy, kde najprv vytvoríme vstup pre model, ďalej definujeme meno, podlá ktorého chceme aby sa natrénovaný model uložil, potom definujeme špeciálnu vrstvu prístupu, a jej parametre, pre jednoduchšie zvolenie parametrov mame tieto možnosti už napísane a sú len zakomentované. Následne vložíme Unet architektúru, a definujeme výstupnú vrstvu. Všetky tieto kroky, využívajú ako vstupný parameter predošlú vrstvu, čo nám umožňuje jednoducho ich spájať. Ku koncu použijeme funkciu define\\_model na vytvorenie modelu. Tieto kroky sú všeobecne pre každý prístup s tým rozdielom, že používame zakaždým inú špeciálnu vrstvu a to podľa prístupu.

\hspace{10mm}Výnimka je u transfer learning prístupu, kde je potrebný import modelu, ktorý chceme použiť pre tento prístup. a teda jeho stiahnutie a následne použitie ako špeciálnu vrstvu a potom nasledujú rovnaké kroky ako u predošlých s tým rozdielom, že sieti ešte definujeme určitý počet vrstiev, pri ktorých môže upravovať naučene váhy a kde nie. Tuto časť programu môžeme vidieť na obrázku č. \ref{impl:tra}

\hspace{10mm}Pri prístupe Gáborové filtre, ešte musíme získať váhy modelu a následne nastaviť nové váhy. Váhy, kde sa už nachádzajú Gáborové filtre. Túto operáciu vykonávame po definovaní modelu.

\hspace{10mm}Po všetkých týchto krokoch mame už model, ktorý môžeme trénovať a ďalej evaluovať a testovať. Nasleduje časť, kde definujeme vlastné metriky pre overenie riešenia. Kompilácia modelu s callbacks, optimalizátor, používame optimalizátor Adam, loss funkciu categorical crossentropy a metriky accuracy, recall, precission, a iné pričom tie sú využívane z frameworku Tensorflow. Po tomto kroku začneme trénovať s príslušnými parametrami a validačnými dátami. Potom uložíme model. a vykonávame evakuáciu aj s reálnou predikciou a vizualizáciou grafov pomocou knižnice Pylab. Po týchto krokoch mame možnosť načítania špecifického modelu, vizualizáciu filtrov v prvej vrstve tohto modela a evaluacií na tomto modeli. Celý kód moste nájsť v prílohe č. \ref{kod}.

Spracovanie obrazových dát metódami umelej inteligencie je v dnešnej dobe jedna z najčastejších oblastí počítačovej vedy, ktorá ide dopredu a, ktorá sa skúma, hľadajú sa nové riešenia, objavujú sa nové poznatky. My sme si zobrali za úlohu skúmať hlboké neurónové siete, konkrétne konvolučné neurónové siete a ich základné časti, teda metódy generovania filtrov. Vybrali sme si rôzne typy metód generovania filtrov, pričom tieto typy vieme jednoducho rozdeliť do dvoch hlavných skupín a to na automatické generovanie filtrov a manuálne generovanie filtrov. Za automatické sme vybrali metódu generovania filtrov pomocou backpropagation, ktoré je klasicky spôsob generovania filtrov, čo sa týka použiteľnosti, používa sa skoro všade a je to štandardný spôsob. Ďalej sme si vybrali generovanie filtrov pomocou autoenkóderu, ktorý svojou podstatou pomáha zjednodušovať vstupy a následne je jednoduchšie nachádzať v jeho výstupe vzory, poprípade určite objekty. Transfer learning ako metóda, ktorá sa stáva viacej modernejšia a používanejšia v dnešnej dobe. A posledná metóda na generovanie filtrov sú filtre vytvorené pomocou Gáborovej funkcie, tato metóda sa radi do kategórii manuálne generovanie filtrov.

Jednotlivé prístupy, metódy, sme sa rozhodli testovať a porovnávať na dátach, ktoré budú pre sieť náročne na rozpoznanie a taktiež na dátach, ktoré pre sieť budú jednoduché nerozoznanie. Vďaka tomu sme chceli zistiť či tieto prístupy sú rozdielne, či majú podobné výsledky, ktorá z týchto metód je lepšia a, ktorá je horšia, a to aj na akom type dát. Ako dáta, ktoré budú pre sieť náročne, sme použili histologické dáta, konkrétnejšie v časti \ref{dataset:pcam}, ako dáta, ktoré budú pre sieť jednoduchšie sú dáta rôznych textúrových povrchov, konkrétnejšie v časti \ref{dataset:dagm}

Rozhodli sme sa jednotlivé prístupy otestovať s viacerými parametrami, ktoré by mohli ovplyvniť výsledky. Pre metódy, Gáborové filtre, backpropagation, autoencoder, sme sa rozhodli meniť parametre, ktoré mohli ovplyvňovať výsledky jednotlivých neúrovňových sieti. Tieto parametre sú veľkosť  filtrov a počet filtrov. Pri metóde transfer learning, sme vybrali rôzne typy modelov a množstvo filtrov, ktoré bude sieť môcť upraviť.

Ako najlepší princíp pre klasifikáciu v histologických dátach je metóda generovania filtrov pomocou Gabrových filtrov, pričom tato metóda sa nám ukázala veľmi nestabilná, čo značí, že pri zmene spomínaných parametrov veľmi kolíše úspešnosť tejto klasifikácie a to až do takej miere, že sieť v niektorých prípadoch zlyhávala, teda úspešnosť klasifikácie bola skoro nulová.

Pričom najstabilnejšie výsledky a zároveň najlepšie dosahuje metóda s využitím backpropagation, tá síce reagovala na zmenu parametrov, ale nie tak drasticky ako Gáborové filtre. Táto metóda dosahovala veľmi dobre výsledky aj v druhom datasete, ale nikdy nedosiahla najlepšie čo značí o jej prispôsobiteľnosti.

Čo sa týka metódy generovania filtrov pomocou autoenkóder, táto metóda dosahovala tiež vhodne výsledky, no v porovnaní s ostatnými boli až 3 v poradí v oblasti histológie. Najlepšie výsledky podávala vtedy, pokiaľ dáta boli veľmi podobného charakteru, čo sme mohli vidieť v dataset DAGM \ref{dataset:dagm}. Tam dosiahla zo všetkých skúmaných najlepšie výsledky, a to pomerne stabilné, teda na úspešnosť zmena jednotlivých parametrov až tak nevplývala.

Čo sa týka metódy s použitím transfer learningu, tato metóda v našich doménach, teda histológii a textúrových povrchov dosiahla katastrofálne výsledky a vôbec nedokázala klasifikovať a ucíť sa. Teda tato metóda je nepoužiteľná, pokiaľ model chceme používať už naučený, ale nie na podobnej doméne.