Semestrální práce
KKY/ZDO 2024

Pavel Balda (A23B0047P)

1 Úvod

Cílem této práce je s pomocí metod počítačového vidění určit počet stehů na obrázcích ze cvičného šití studentů medicíny.

K dispozici máme 134 různě velkých obrázků stehů s jejich anotacemi. Anotace jsou k dispozici ve formě *XML* souboru s uloženými souřadnicemi bodů lomené čáry rány (incision) a souřadnicemi dvou bodů úsečky jednotlivých stehů (stitches). Množina obsahuje i nerozhodnutelná data.

Základním problémem v této úloze je rozlišit steh od nestehu (místa v ráně, kde je naznačeno, že by steh být mohl, nicméně není) a určit počet stehů na obrázku. Jedná se tedy z podstaty o úlohu klasifikace do více tříd. Algoritmus by měl navíc umět rozhodnout o nerozhodnutelnosti (brát ji jako jednu ze tříd).

Během vypracování jsem postupoval samostatně a své metodické řešení jsem s nikým nekonzultoval. Vycházel jsem z materiálů k předmětu KKY/ZDO a internetových zdrojů a dokumentací. K psaní kódu a debugování jsem využíval ChatGPT 3. Parsovací funkci argumentů jsem převzal od Pavla Březiny, jemuž za to náleží veliký dík.

Práce se skládá z repozitáře na adrese `https://github.com/pavbal/ZDO_SP` a této dokumentace. V repozitáři jsou k dispozici zdrojové kódy a datasety. Ze zdrojových kódů zde jsou jak nepostradatelné, tak ty použité při trénování neuronových sítí (ty z podstaty nejsou potřebné, když již máme model).

2 Vypracování

2.1 Předzpracování

Rozhodl jsem se pro účely práce vycházet z šedoténového obrázku, neboť hlavní informaci o problematice nesou rozdíly celkového jasu a nikoliv barev.

2.1.1 Horizontalizace rány

Nejprve jsem se soustředil na narovnání roviny rány (horizontalizaci). Vycházím z předpokladu, že úhel rány v obrázku se neliší o víc jak 10° od vodorovné osy obrázku.

Mé pojetí horizontalizace vychází z dalšího předpokladu, že oblast rány má výrazně menší (resp. větší, v případě použitého negativu) jasové hodnoty než okolí rány. Posledním předpokladem je tvarově lineární charakter rány.

Měnil jsem tedy náklon obrázku (prováděním rotací) v rozmezí +- 10° a použil jsem to pootočení, pro které bylo největší maximum histogramu ze součtu horizontálních hodnot negativu (tj. rána byla v rovině). Získání tohoto stupně mi následně umožnilo správně pootočit původním obrázkem. Tento obrázek jsem následně oříznul od vzniklých nulových hodnot.

Toto oříznutí samozřejmě může poškodit informaci, která by mohla být důležitá pro správnou klasifikaci a detekci stehů. Usoudil jsem však, že přínos narovnání rány je vzhledem k povaze

dat a dalšímu zpracování cennější než případná ztráta části informace.

Postup je znázorněn v prvním sloupci obrázku 1.

2.1.2 Segmentace rány a oříznutí stran

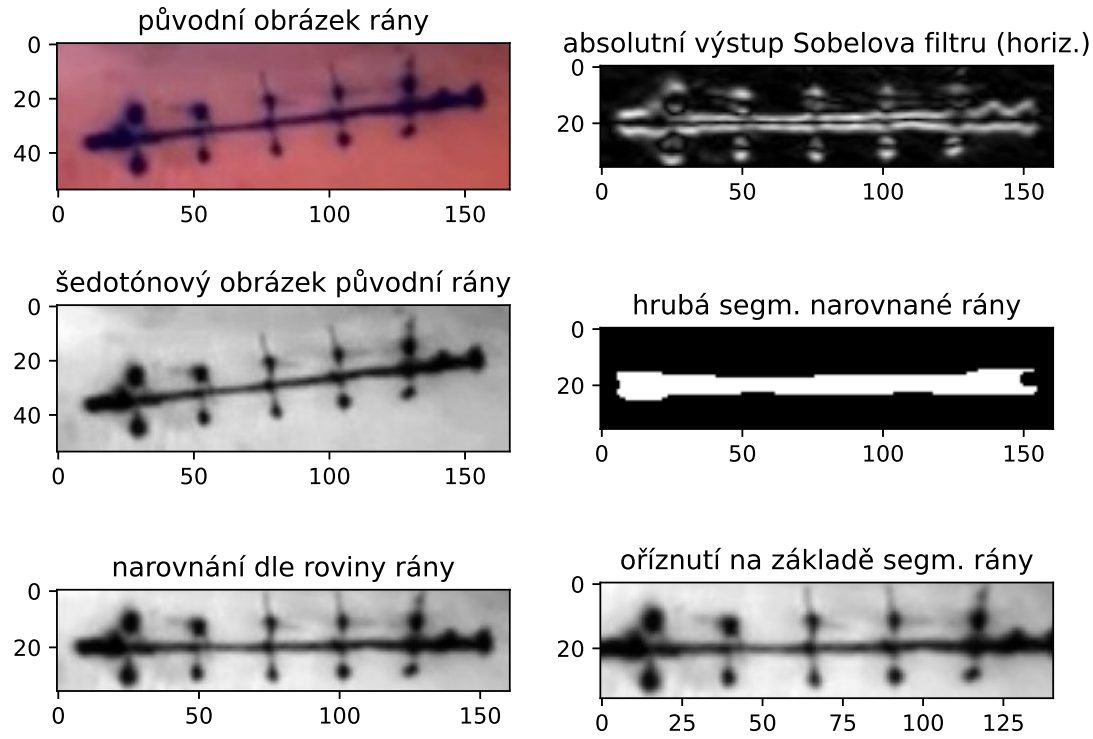
Teď když máme narovnanou ránu, provedeme její přibližnou segmentaci. K tomu využijeme Sobelův filtr detekující horizontální hrany. Protože máme nyní ránu v rovině (s protože stehy jsou na ránu přibližně kolmé), dokážeme Sobelovým filtrem efektivněji určit polohu rány a minimalizovat vliv stehů.

Absolutní výstup Sobelova filtru jsem následně vyprahoval dle automaticky zvoleného prahu. Získal jsem tak binární obrázek, ze kterého jsem sérií metod matematické morfologie (uzavření, otevření, vyplnění děr, eroze) a použitím vhodných jader (kernelů) získal přibližnou segmentaci rány.

Účelem této segmentace bylo najít začátek a konec rány (podle nejmenší a největší x-ové souřadnice objektu v binárním obrázku). Po zjištění těchto souřadnic jsem šedotónový obrázek rány na vstupu této metody ořízl ze stran (sebral jsem o pár pixelů víc, abych se zbavil náběžných hran rány kvůli následujícímu předzpracování).

Na výstupu této metody je tedy šedotónový obrázek rány natočený a oříznutý ze stran. Rána by tedy měla vést horizontálně po celé horizontální délce obrázku.

Postup je znázorněn ve druhém sloupci obrázku 1.



Obrázek 1: Znázornění postupu předzpracování. Nejprve narovnání (vlevo), následně oříznutí ze stran (vpravo). Vývoj shora dolů, zleva doprava (po sloupcích). Výsledek dané úpravy je vždy nejníže.

2.1.3 Detekce stehů a nestehů

Nyní se budeme věnovat detekci míst, kde se nacházejí stehy.

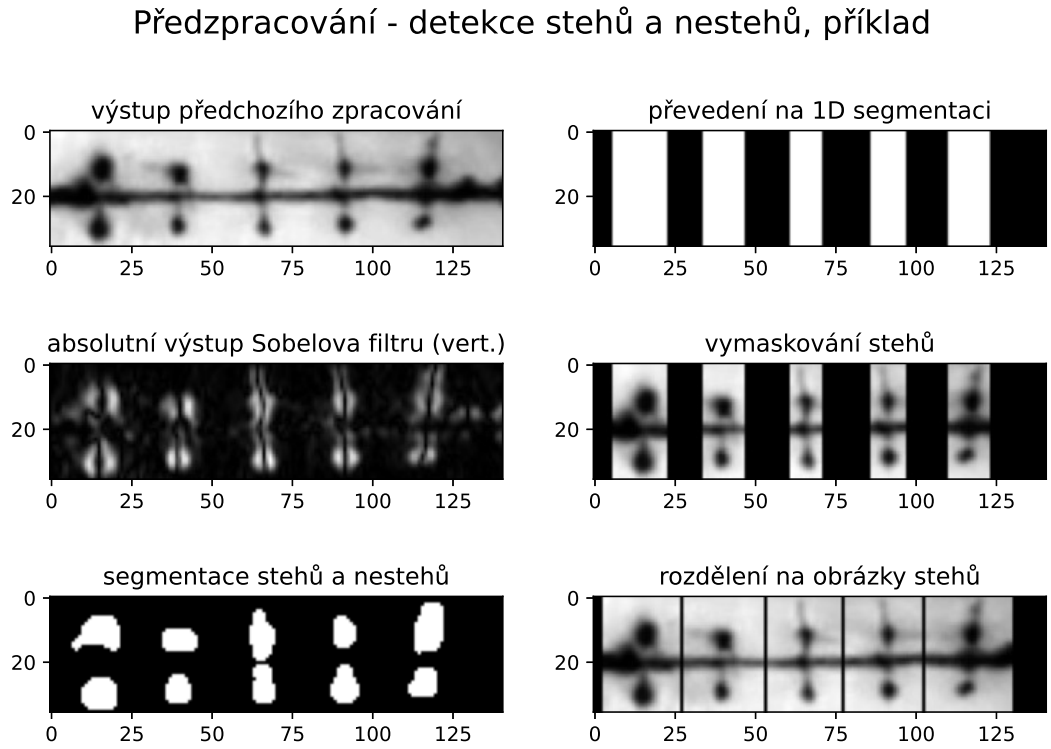
Pro to jsem využil absolutní výstup tentokrát vertikálního Sobelova filtru. Ten jsem vyprahoval (práh jsem urči z histogramu tak, aby byl zachován procentuální poměr objekt:pozadí v obraze) a sérií metod matematické morfologie jsem tento získaný binární obraz převedl na hrubou segmentaci stehů a nestehů. Často jsou segmentovány právě nakreslené body vyznačující začátek a konec potenciálních stehů. Protože pro budoucí použití mě zajímá hlavně horizontální hranice stehů, převedl jsem 2D segmentaci na 1D segmentaci. Tím jsem získal jakési přibližné bounding-boxy jednotlivých stehů a nestehů.

S využitím těchto bounding-boxů bychom mohli určit jednotlivé stehy pro extrakci příznaků a další klasifikaci. Bohužel jsou však tyto bounding-boxy příliž nekonzistentní ve své šířce a mnoho z nich nedokáže zachytit nitky vedoucí z uzlíků, které osobně hodnotím jako nejvýpovědnější příznakovou oblast jednotlivých stehů.

Abych tento nedostatek omezil, rozšířil jsem bounding-boxy stehů a nestehů až do poloviny mezer mezi nimi. Tím se stalo to, že byl celý obrázek rozsekán na jednotlivé stehy a ztratí se nám tak méně informace. Navíc je tím zaručen konzistentnější poměr stran stehů v budoucím

datasetu.

Postup je znázorněn na obrázku 2.



Obrázek 2: Znázornění postupu předzpracování. Nejprve segmentace stehů (vlevo), následně rozdělení obrázku na úseky jednotlivých (ne)stehů (vpravo). Vývoj shora dolů, zleva doprava (po sloupcích).

2.2 Tvorba datasetu stehů a nestehů

Nyní máme tedy narovnaný a oříznutý obrázek rány vertikálně rozdělený na jednotlivé stehy a nestehy.

Nyní je třeba zajistit jejich anotaci. Ze *XML* souboru jsem si extrahoval souřadnice jednotlivých stehů. Vytvořil jsem si nulovou (černou) kopii úplně původního "raw"obrázku rány (ještě před předzpracováním), k nimž anotace sedí, a do této kopie jsem vykreslil začátky a konce stehů jako bodové objekty. Na tento binární obrázek jsem následně použil stejnou rotaci a stejné oříznutí jako na původní obrázek v předzpracování. Podle souhlasu 1D segmentace a výskytu nenulových prvků na souhlasných pozicích jsem poté určil, v jakém případě se jedná o stehy a v jakém o nestehy.

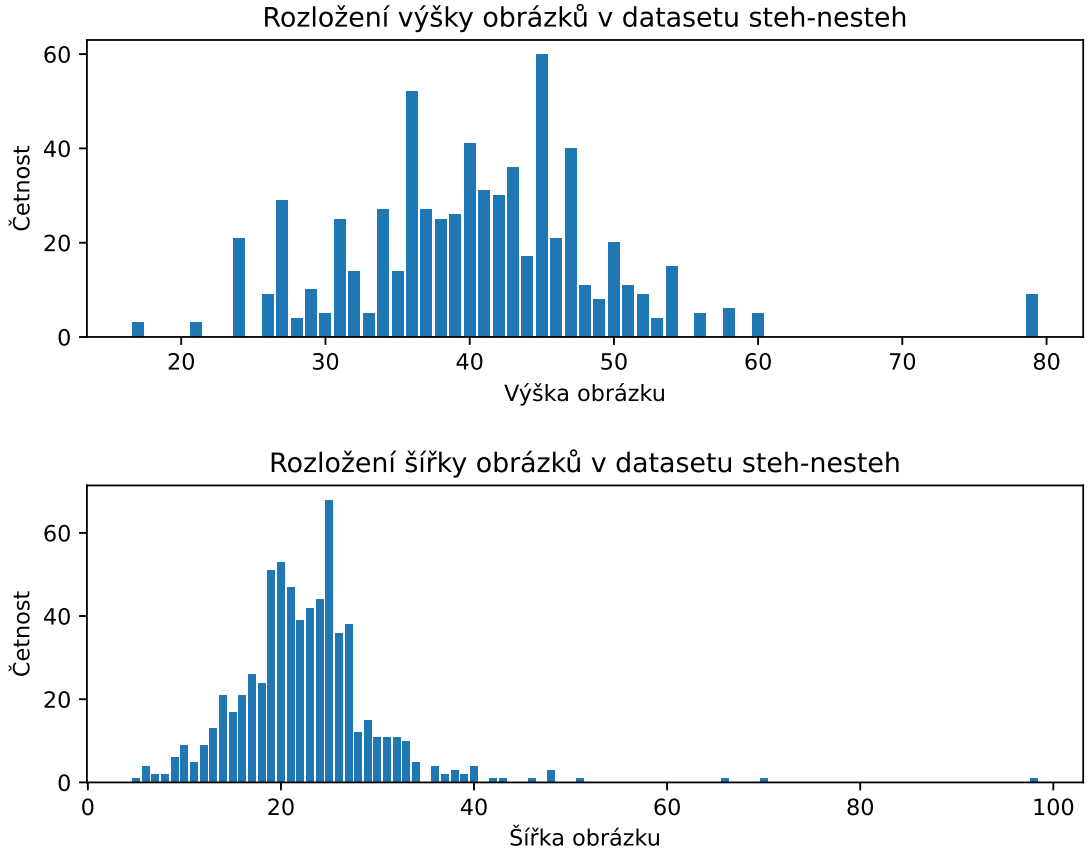
Tím jsem získal anotace pro každý steh a nesteh získaný předzpracováním a de facto vytvořil klasifikační dataset o dvou třídách - steh a nesteh. Dalo by se tedy říci, že jsem problém klasifikace do více tříd zjednodušil na klasifikaci do třídy jedné.
Tento postup s sebou nese určité nevýhody. K největší nevýhodě patří ztráta kontextu mezi stehy na stejné ráně. Tuto nevýhodu jde jen velmi omezeně kompenzovat v postprocessingu

(viz závěr). Zjednodušení úlohy však také může přinést hodně výhod. Mezi ty zásadní patří převod na dichotomii (steh-nesteh) a také zvětšení počtu trénovacích dat. Budeme tedy řešit jednodušší úlohu s více dostupnými daty.

2.2.1 Vlastnosti datasetu steh-nesteh

Vzniklý dataset čítá 678 obrázků, z čehož 349 náleží třídě nesteh (0) a 329 náleží třídě steh (1). Toto rovnoměrné zastoupení tříd je pro úlohu dichotomie ideální. Výškový rozměr (y) obrázků se pohybuje mezi 17 a 79 pixely. Šířkový rozměr se pohybuje mezi 5 a 98 pixely. Medián a průměr vertikálního rozměru jsou přibližně stejné a činí cca 40 pixelů. Průměr a medián šířky jsou taktéž téměř stejné a činí cca 22 pixelů. Detailnější rozložení můžeme vidět na obrázku 3.

Tyto údaje nám značí, že dataset sice není příliš konzistentní, co se rozměrů obrázků týče, avšak pomineme-li outliery, dostaneme se na příznivější rozptyl pro práci s datasetem a klasifikačními algoritmy.



Obrázek 3: Vizualizace rozložení rozměrů obrázků v datasetu.

2.3 Klasifikace

Neboť nyní máme nyní více dat, rozhodl jsem se klasifikaci realizovat pomocí jednoduché konvoluční neuronové sítě. Dohromady jsem natrénoval cca 25 modelů. Zkoušel jsem především dvě různé architektury - první byla menší (co se počtu neuronů týče), druhá objemnější (ta

také dosahuje obecně lepších výsledků).

2.3.1 Architektura menší sítě

Struktura vrstev sítě vypadá následnově (popořadě):

- První konvoluční vrstva o 8 filtrech a velikosti jádra 3.
- Max-pooling vrstva zmenšující s velikostí jádra 2 a krokem 2.
- Druhá konvoluční vrstva o 16 filtrech a velikostí jádra 3.
- Max-pooling vrstva obdobná té předchozí.
- První fully-connected z 800 neuronů předchozí vrstvy na 64 neuronů.
- Druhá fully-connected z 64 neuronů do jednoho neuronu.

Jako aktivační funkci používá síť ReLU. Na výstupní vrstvě pak používá sigmoidální funkce, která se dá využít, neboť se jedná o úlohu dichotomie.

Důvodem pro zkoušení této architektury byla možnost, že by se objemnější model mohl přetrénovat a overfitoval se na trénovací data. Tato pochybnost se ukázala spíše jako neopodstatněná.

2.3.2 Architektura objemnější sítě

Struktura vrstev sítě objemnější vypadá následnově (popořadě):

- První konvoluční vrstva o 16 filtrech a velikosti jádra 3.
- Max-pooling vrstva zmenšující s velikostí jádra 2 a krokem 2.
- Druhá konvoluční vrstva o 32 filtrech a velikostí jádra 3.
- Max-pooling vrstva obdobná té předchozí.
- První fully-connected z 1600 neuronů předchozí vrstvy na 128 neuronů.
- Druhá fully-connected z 128 neuronů do 2 neuronů.

Jako aktivační funkci používá síť ReLU. Na výstupní vrstvě pak síť uplatňuje funkci Softmax.

Modely této architektury dosahovaly mnohem lepších výsledků než předchozí. Další odstavce i finální řešení tedy uvažují použití právě této strukruty modelu.

2.3.3 Transformace

Před vstupem obrázků do sítě je nutné (nebo žádoucí) je transformovat. Nejprve převádím všechny obrázky na stejný rozměr (40 na 20 pixelů), pak je převedu na tenzor a normalizuji kolem hodnoty 0 (interval od -0,5 do +0,5).

Mezi transformace pro účely trénování mám také zařazeny augmentace, abych zajistil vetší variablitu dat. Používám náhodné vodorovné a svislé překlopení a náhodné otočení o 180 stupňů, neboť jsou to operace, které zachovávají povahu a podstatu stehů tak, jak jsou vybírány předzpracováním a jak jsou zobrazeny v původním i novém datasetu.

2.3.4 Trénování a výběr modelu

.
Abychom mohli efektivně trénovat, potřebujeme validační množinu. Tu jsem náhodně vybral jako pětinu našeho datasetu. Pro účely ladění používám celou dobu tuto jednu zvolenou podmnožinu. Zbytek datasetu je vyhrazen na přímé trénování.

Jako ztrátovou funkci používám *CrossEntropyLoss()*. Jako optimizer volím *Adam* s konstantou učení 0,001. Hodnotu batch_size jsem zvolil 16 (zkoušel jsem 4, 8 a 16).
V průběhu trénování monitoruji přesnost (accuracy) a ztrátu (loss) na validační množině a ukládám pouze modely s nejlepším výsledkem těchto dvou metrik.

S tímto nastavením se přesnost (accuracy) dostala během trénování až na 91 % na validačních datech.

Dle vývoje ztráty a přesnosti, které dosahovali nejlepších výsledků vždy kolem 800. epochy, jsem se rozhodl natrénovat model na 800 epoch a to tentokrát z celého datasetu a bez validace. Je tak zajištěna větší variabilita dat a je tedy pravděpodobné, že výsledný model bude mít lepší generalizační schopnost.

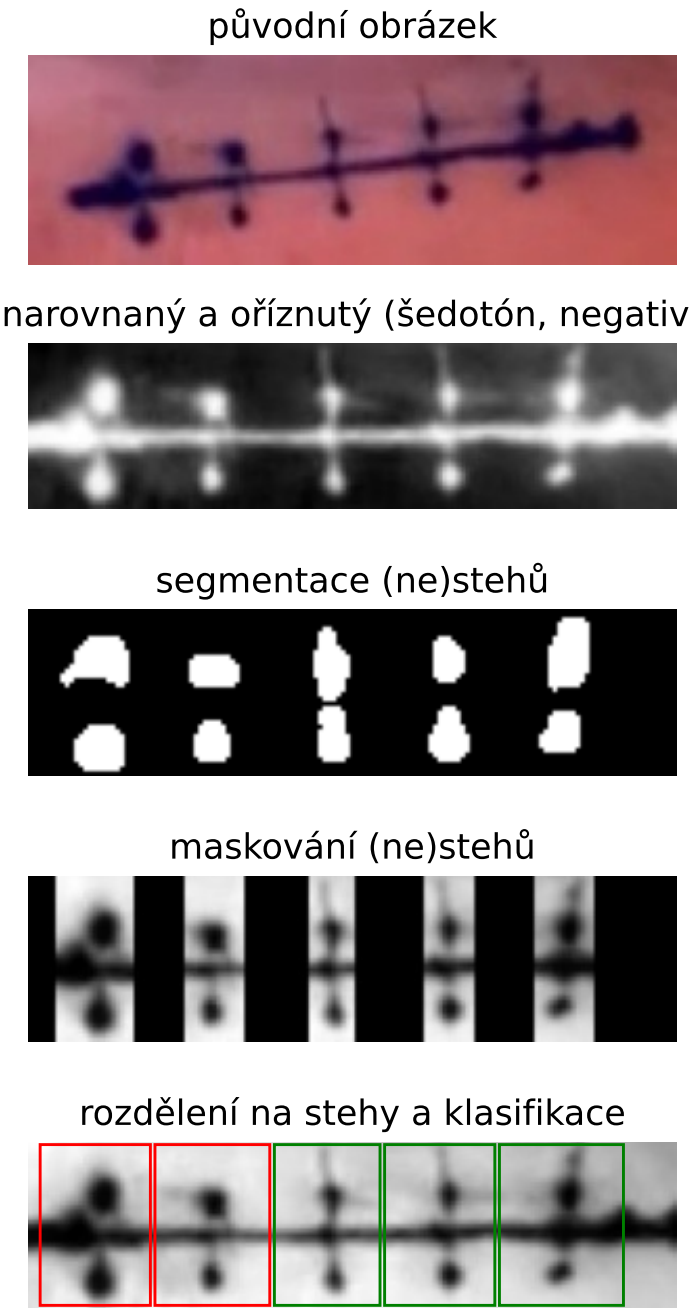
2.4 Shrnutí procesu určení počtu stehů

Vstupní obrázek tedy nejprve projde předzpracováním, kde bude správně natočen, oříznut a rozsekán na jednotlivé (ne)stehy. Ty následně po jednom přijdou na vstup neuronového modelu, který každý z nich přidělí do třídy 1 (steh), nebo třídy 0 (nesteh). Sečtením těchto příslušností získáme odhad počtu stehů na vstupním obrázku.

2.5 Vizualizace a funkcionalita

v Github repozitáři se dle zadání nachází skript *run.py*, který by měl fungovat dle zadaných kritérií. Skript má možnost vizuálního módu (viz 4), kdy se za sebou zobrazí ilustrace postupu algoritmu a výsledná klasifikace jednotlivých stehů (zelená - steh, červená - nesteh, bez ohraničení - nebráno v potaz klasifikátorem). Jedná se o přibližnou vizualizaci, výřezy puštěné do klasifikátoru mají lehce odlišný rozměr.

Aby skript dobře fungoval, je třeba jako argument použít relativní cestu k obrázkům ze složky *src*. Výstupní soubor se pak uloží do souboru zvoleného jména do složky *src*. Nedoporučuji při vizuálním módu vkládat jako argument příliš mnoho obrázků (kvůli nutnosti zavírat okna).



Obrázek 4: Vizualizace postupu a predikce. Zobrazení ilustračně významných kroků algoritmu včetně výsledné predikce (dole). Počet zelených bounding-boxů je roven predikci počtu stehů pro vstupní obrázek.

3 Závěr

V této práci jsem se zabýval problémem klasifikace do více tříd, které odpovídají počtu zobrazených stehů. Tento problém jsem s použtím metod probíraných na přdnáškách a cvičeních převedl na jednodušší problém úlohy dichotomie, pro kterou jsem natrénoval a použil konvoluční neuronový model.

Výhody mého přístupu spočívají v potenciálně větší robustnosti z hlediska klasifikace jednotlivých stehů, neboť je k dispozici více trénovacích dat na typově jednodušší úlohu. Další výhodou mého přístupu je možnost využití meziproduktu algoritmu a jeho použití např pro úlohu detekce stehů, případně jejich vymaskování apod.

Hlavní nevýhodou mého modelu je fakt, že nedokáže klasifikovat nerozhodnutelná data (obrázky, ze kterých nelze určit počet stehů z jakéhokoliv důvodu). Další nevýhodou je fakt, že řešení nezohledňuje vzájemný vztah mezi jednotlivými (ne)stehy na jedné ráně.

Oblasti možného zlepšení přímo vyplývají z hlavních nevýhod. Nerozhodnutelné obrázky by mohly být odfiltrovány již během předzpracování, případně by se dal natrénovat jednoduchý klasifikátor (například na bázi HOG), který by dokázal rozpoznat nerozhodnutelná data od klasifikovatelných. Co se týče vzájemného vztahu mezi stehy, šel by algoritmus upravit způsobem, že by porovnával výstupy neuronového modelu stehů vedle sebe (neboť stehy jsou téměř vždy vedle sebe) a dělal případnou korekci v případě obklíčení klasifikovaného nestehu dvěma klasifikovanými stehy.

Tato práce mě bavila a byla mi velkým přínosem, neboť jsem si vyzkoušel aplikaci použitých metod včetně aplikace konvolučních neuronových sítí, kterými se dozajista budu zabývat ve své diplomové práci.