VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV BIOMEDICÍNSKÉHO INŽENÝRSTVÍ
DEPARTMENT OF BIOMEDICAL ENGINEERING

SEGMENTACE INTRAKARDIÁLNÍCH ZÁZNAMŮ

SEGMENTATION OF INTRACARDIAL ECG

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Jan Řehoř
AUTHOR

VEDOUCÍ PRÁCE Ing. Petra Novotná
SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Bioinženýrství**

Ústav biomedicínského inženýrství

Student: Bc. Jan Řehoř ***ID:*** 195747

Ročník: 2 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Segmentace intrakardiálních záznamů

POKYNY PRO VYPRACOVÁNÍ:

1) Vytvořte literární rešerši se zaměřením na zpracování intrakardiálních záznamů, se zaměřením na identifikaci diagnosticky výtěžných úseků záznamu a extrakci síňové (případně komorové) aktivity. 2) Seznamte se s databází IECG dostupné na ÚBMI a proveďte potřebné kroky předzpracování, včetně manuální anotace. 3) Navrhněte a realizujte model automaticky segmentující IECG záznam (viz bod 1) zadání). Model realizujte ve vhodném programovacím prostředí (Python, Matlab). 4) Proveďte segmentaci na dostupné databázi a výsledky vyhodnoťte v kontextu manuálně vytvořených anotací. 5) Na základě výsledků z předešlých bodů zadání optimalizujte nastavení modelu. 6) Dosažené výsledky podrobně interpretujte a diskutujte. Proveďte porovnání s dostupnými zahraničními studiemi na podobné téma.

DOPORUČENÁ LITERATURA:

[1] Zanon F. et al. Basic Properties And Clinical Applications Of The Intracardiac. J Atr Fibrillation, 9(4):1444, 2016, doi:10.4022/jafib.1444.

[2] Alhusseini, M. I. et al. Machine Learning to Classify Intracardiac Electrical Patterns During Atrial Fibrillation. Circulation: Arrhythmia and Electrophysiology, 13(8), 2020, ISSN 1941-3149.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 20.5.2022

Vedoucí práce: Ing. Petra Novotná

doc. Ing. Radim Kolář, Ph.D.
předseda rady studijního programu

Abstrakt

Tato diplomová práce se zabývá segmentací intrakardiálních záznamů EKG a je rozdělena do několika částí. První část je spjata s teoretickým seznámením problematiky, tedy například jak funguje srdce, co je intrakardiální EKG a konvoluční neuronová síť. Další části práce jsou již tvořeny praktickou částí, tedy, anotací signálu a návrhu modelu automaticky segmentující intrakaridální záznam. Po praktickou částí pokračuje zhodnocení výstupů z řešení, porovnání s řešením třetí stran a se zahraničními studiemi věnující se podobnému tématu. Poslední část práce tvoří diskuze a závěr, kde jsou shrnuty výsledky práce.

Klíčová slova

Intrakardiální EKG, anotace signálu, segmentace intrakardiálního EKG, konvoluční neuronová síť, strojové učení.

Abstract

This master´s thesis deals with the segmentation of intracardiac ECG recordings and is divided into several parts. The first part is connected with a theoretical acquaintance with the issue, such as how the heart works, what is an intracardiac ECG and a convolutional neural network. Other parts of the work are already formed by the practical part, ie, signal annotation and model design automatically segmenting the intracaridal record. After the practical part, the evaluation of the results of the solution continues, comparison with the solution of third parties and with foreign studies dealing with a similar topic. The last part of the work is a discussion and conclusion, which summarizes the results of the work.

Keywords

Intracardiac ECG, signal annotation, intracardiac ECG segmentation, convolutional neural network, machine learning.

Bibliografická citace

Řehoř, J. *Segmentace intrakardiálních záznamů*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav biomedicínského inženýrství, 2022. 94 s., 15 s. příloh. Diplomová práce. Vedoucí práce: Ing. Petra Novotná.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	*Bc. Jan Řehoř*
VUT ID studenta:	*195747*
Typ práce:	*Diplomová práce*
Akademický rok:	*2021/22*
Téma závěrečné práce:	*Segmentace intrakardiálních záznamů*

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

.

V Brně dne: 20. května 2022 ----------

podpis autora

Poděkování

Děkuji vedoucí diplomové práce Ing. Petře Novotné za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce. Dále bych chtěl poděkovat Ing. Jakubu Hejčovi za cenné rady v problematice anotací signálů a Ing. Richardu Ředinovi za přispění dat při anotaci.

V Brně dne: 20. května 2022 ----------

podpis autora

Obsah

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Vyšetřovací metody srdce ušly za dobu existence lidstva obrovský kus cesty. Díky nim lze v dnešní době získat o funkčnosti srdce důležité informace bez potřeby explorativní chirurgie, která je zátěží nejen pro samotného pacienta, ale i pro lékařský personál. Při těchto vyšetřovacích metodách je třeba se spoléhat na projevy srdce, které lze nějakým způsobem měřit. Jeden z těchto projevů, elektrickou aktivitu, se snažil na počátku 20. století zaznamenat nizozemský fyziolog Willem Einthoven, vynálezce velmi známého elektrokardiografu (EKG). [1] Díky němu se neinvazivní vyšetřování srdce posunulo o velký kus dále a jelikož se jednalo o nenáročné zařízení, získalo si rychle popularitu.

Ačkoliv je povrchové EKG neinvazivní, vlivem útlumu skrze tkáň ztrácí signál část své informace. Za určitých okolností je tedy potřeba měření provést invazivně, proto se lze setkat i s intrakardiálním EKG, které měří signál přímo v srdci.

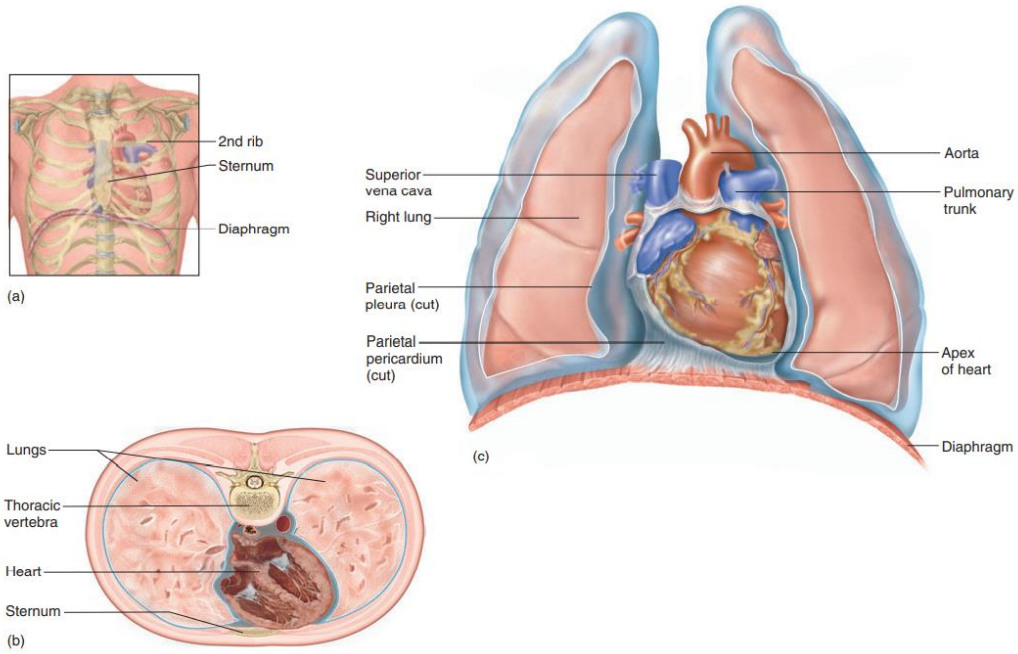
Z povrchového, nebo intrakardiálního EKG signálu je třeba získat potřebné informace. Může se jednat například o segmentaci, určení tepové frekvence nebo stanovení arytmií. Dříve tyto informace získával pouze lékař, nicméně s rozvojem algoritmů strojového učení jsou programy čím dál tím více schopné tento proces zvládat samy.

Cílem této práce je segmentace intrakardiálních záznamů. První část tvoří teoretické seznámení problematiky, tedy co je srdce a jak funguje, co je intrakardiální EKG a čím se liší od EKG povrchového. Na závěr této části bude věnována pozornost konvoluční neuronové síti. Další část bude již praktická, bude probráno jak a pomocí jakého nástroje byla provedena ruční anotace intrakardiálního EKG signálu a realizace modelu automaticky segmentující intrakardiální EKG signál.

1. TEORETICKÝ ÚVOD

1.1 Srdce

Lidské srdce je svalový orgán tvaru nepravidelného kužele situovaný v mediastinu mezi pravou a levou plící za hrudní kostí z 2/3 vpravo (obrázek **1.1**). Hmotnost srdce dospělého jedince se pohybuje kolem 300 g. Je obalen vrstvou zvanou perikard (osrdečník), který je ukotven na bránici a pojivové tkáni velkých cév nad srdcem. [2][3]



Obrázek 1.1 Pozice srdce v hrudní dutině. (a) Pozice srdce vůči hrudnímu koši. (b) Transverzální řez hrudníku v úrovni srdce. (c) Frontální řez hrudní dutiny [2]

Obsahuje čtyři dutiny: pravou předsíň (atrium dextrum), levou předsíň (atrium sinistrum), pravou a levou komoru (ventriculus dexter et sinister) oddělené chlopněmi. Srdce je navíc rozděleno přepážkou na pravou a levou část, kdy v každé části je jedna předsíň a komora. Předsíně jsou tenkostěnné a přitéká do nich krev z žil. Komory disponují naopak silnou svalovou stěnou a na rozdíl od předsíní, do kterých krev přitéká, slouží k vypuzování krve do tepen. [4] Chlopně slouží k jednosměrnému toku krve. Mezi pravou předsíní a komorou je takzvaná trojcípá chlopeň, která je tvořena třemi cípy. Na levé straně odděluje předsíň a komoru chlopeň dvojcípá, někdy také nazývána jako mitrální. Mezi komorami a velkými tepnami se nachází chlopně semilunární a to konkrétně pulmonární (mezi pravou komorou a plicnicí) a aortální (mezi levou komorou a aortou). [2]

Srdce je z funkčního hlediska tvořeno dvěma typy buněk. Buňkami převodního systému, které jsou schopné samovolné elektrické aktivity a buňkami pracovního myokardu, které odpovídají na elektrickou aktivitu stahem. [5][6] Tato skutečnost nabízí alespoň stručné zmínění charakteristických vlastností srdce, a to autonomii a automacii[1]. Autonomii lze chápat jako schopnost srdce vytvářet vzruchy a stahy bez závislosti na centrální nervové soustavě. Díky automacii je srdce pak schopné samočinného a opakovaného střídání stahů (systoly) a relaxace (diastoly) pracovního myokardu [3][7].

Za fyziologického stavu tak dochází pravidelně k jednotlivým fázím srdečního cyklu, a to k izovolumetrické kontrakci, ejekční fázi, izovolumetrické relaxaci a plnící fázi. Izovolumetrická kontrakce je první fáze komorové systoly, kdy roste tlak a tím, že semilunární chlopně jsou uzavřené a krev se nestlačuje, dochází k uzavření cípatých chlopní. V ejekční fázi nastává překročení nitrokomorového tlaku nad diastolický tlak v tepnách a dochází k otevření semilunárních chlopní a vypuzení krve do tepen. Z pravé komory je vypuzena do plicnice neokysličená krev a z levé komory okysličená krev do aorty. Následuje izovolumetrická relaxace. Nitrokomorový tlak postupně klesá a dosažením tlaku velkých tepen se uzavírají semilunární chlopně. Poklesem nitrokomorového tlaku na hodnoty nižší, než tlaku síní dochází k otevření atrioventrikurálních chlopní – dvoj a trojcípé. V poslední fázi plnící jsou komory plněny krví z předsíní, která se v nich nahromadila během komorové systoly, otevřením dvojcípé a trojcípé chlopně. V levé předsíni se jedná o okysličenou krev z plicních žil, v pravé naopak neokysličenou krev z horní a dolní duté žíly. Plnění komor je zpočátku rychlé, zvyšuje se tlak a tím plnění zpomaluje, dochází k tzv. diastáze. Nakonec dochází k systole předsíní, které se ale na plnění komor podílí jen z 8 %. [5][6] Díky těmto pravidelným stahům a ochabováním je zajištěn oběh krve a tím i distribuce a výměna kyslíku, výživy a jiných tělu prospěšných i odpadních látek [8].

1.1.1 Akční potenciál

Jak již bylo zmíněno, buňky převodního systému jsou schopné samovolné elektrické aktivity, jinak řečeno, disponují schopností měnit membránový potenciál v buňce.

Klidový membránový potenciál buňky myokardu se pohybuje kolem -50 až –90 mV dle toho, zda se jedná o buňku převodního systému nebo o pracovní myokard. Pro oba typy buněk nicméně vyplývá, že jejich intracelulární prostředí je elektronegativní oproti prostředí extracelulárnímu. [1] Tento potenciál je způsoben nerovnoměrným rozložením iontů v intra – a extracelulárním prostředí buňky. V intracelulárním prostředí je dominantním iontem K+, jehož koncentrace je 30x větší než v extracelulárním prostředí, zatímco vně je vyšší koncentrace Na+ a Ca2+. Jelikož je propustnost membránou buňky

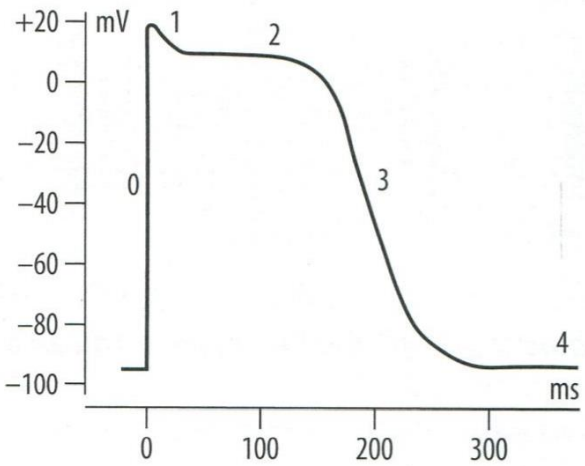
[1] Autonomie a automacie nejsou jediné vlastnosti srdce. Lze zmínit ještě například inotropii, což je schopnost ovlivnění síly srdeční kontrakce, chronotropii, což je schopnost ovlivnění srdeční frekvence anebo také bathotropie neboli ovlivnění prahu vzrušivosti myokardu [6]

pro K+ iont velká, a naopak pro Na+ malá, dochází k nadbytku kladných iontů v extracelulárním prostředí. [6]

Změnou membránového potenciálu nastává akční potenciál, který je manifestován depolarizací, kdy se membránové napětí dostává ze záporných hodnot do kladných a následně dochází k návratu do klidové úrovně. tedy repolarizaci. [5]

Jednotlivé fáze akčního potenciálu jsou trochu různé pro buňky pracovního myokardu a buňky převodního systému. V následujících odstavcích budou popsány více.

V případě buňky pracovního myokardu je klidový potenciál okolo –90 mV. Drážděním buňky začíná akční potenciál depolarizací (otevřením, a tím pádem zvýšením propustnosti, Na+ kanálů), kdy se vnitřek buňky rapidně dostává z hodnoty –90 mV na +20 až +25 mV. Po depolarizaci dochází k částečné, ale rychlé, repolarizaci. Částečná repolarizace, jelikož nedochází k úplné repolarizaci, ale pouze k poklesu zhruba na +10 až +15 mV. Akční potenciál pokračuje fází plató, kdy buňka setrvává po dobu 200 až 350 ms v depolarizovaném stavu. Nakonec dochází k návratu klidového potenciálu, nastává tedy repolarizace. Tento vzruch je označován také jako akční potenciál s rychlou depolarizací. [6]



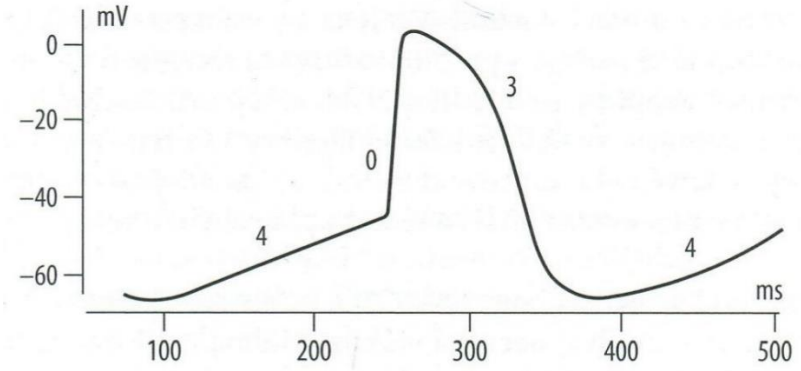
Obrázek 1.2 Akční potenciál buňky myokardu. 0 - depolarizace, 1 - částečná repolarizace, 2 - fáze plató, 3 - pozdní repolarizace, 4 - výchozí úroveň membránového potenciálu [5]

U buněk převodního systému, konkrétně sinoatrialního uzlu (dále SA uzel) a atrioventruklárního uzlu (dále jako AV uzel) je klidový potenciál kolem –50 až -70 mV. Jak již bylo zmíněno výše, tyto buňky jsou schopné samovolné elektrické aktivity. Jeho klidová hodnota se postupně zvyšuje, jedná se o takzvanou spontánní diastolickou depolarizaci, kdy po překročení prahové hodnoty dochází k akčnímu potenciálu, která začíná depolarizací, která je nicméně pomalejší oproti akčnímu potenciálu pracovního myokardu. Jedná se tedy o akční potenciál s pomalou depolarizací. Chybí zde fáze částečné repolarizace a fáze plató. Po depolarizaci tudíž rovnou dochází k repolarizaci,

kdy po návratu do klidového potenciálu nastává opět spontánní diastolická depolarizace a celý proces se opakuje. [5][6]

Jako prevence proti zpětnému šíření vzruchů, dochází od začátku depolarizace až do dvou třetin repolarizace k takzvané absolutní refrakteritě, tedy, úplné nedráždivosti, jelikož většina sodíkových kanálů jsou inaktivní. Ve zbylé třetině repolarizace dosahuje membránový potenciál takových hodnot, že jsou sodíkové kanálky čím dál tím více schopné se otevřít a je možné vyvolat podráždění s čím dál tím menším nadprahovým podmětem. Jedná se o takzvanou relativní refrakterní fázi. Teprve až po repolarizaci je buňka schopná vyvolat akční potenciál. [6]

Akční potenciál pracovního myokardu je zobrazen na obrázku **1.2** a akční potenciál převodního systému na obrázku **1.3**.



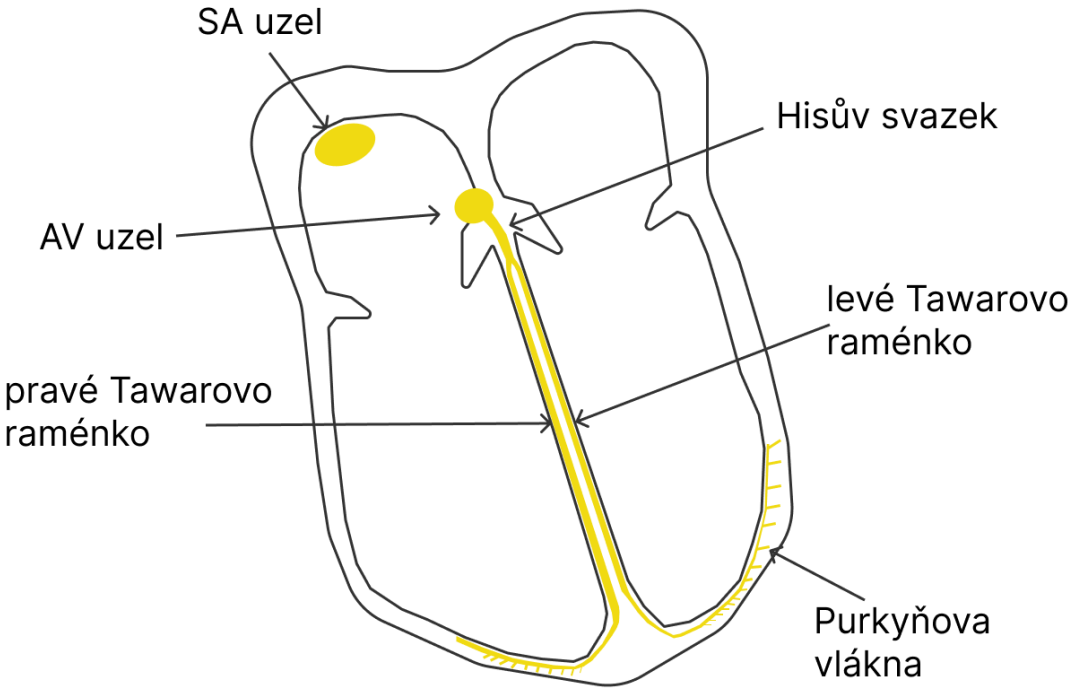
Obrázek 1.3 Akční potenciál buňky převodního systému, 0 – pomalá depolarizace, 3 – repolarizace, 4 – diastolická depolarizace [5]

1.1.2 Převodní systém srdeční

V předchozím textu, který se věnoval problematice akčního potenciálu, byly zmíněny buňky převodního systému srdečního. Převodní systém srdeční lze chápat jako systém buněk schopných elektrické aktivity srdeční. Díky této aktivitě jsou jednotlivé části schopné spontánní depolarizace, avšak za fyziologických podmínek tuto úlohu zajišťuje SA uzel, který slouží jako přirozený pacemaker. Generuje okolo 70 vzruchů za minutu. Tento rytmus je označován jako sinový. Vzruch se šíří do síně a přes internodální síňové spoje (Bachmann, Thorel, Wenckebach) se dostává do AV uzlu a pokračuje do komor přes Hisův svazek, který se rozděluje na pravé a levé Tawarovo raménko, která přechází na Purkyňova vlákna. [5][6]

Za fyziologických podmínek generuje vzruchy SA uzel, nicméně, někdy tuto úlohu může převzít AV uzel, který je pomalejší. Generuje circa 40 až 60 vzruchů za minutu. Nejedná se však o sinový rytmus, ale o takzvaný rytmus nodální. AV uzel také někdy označován jako sekundární pacemaker. Pokud však úlohu pacemakeru převezme jiná

oblast komor, jedná se o rytmus idioventrikulární. [6] Převodní systém srdeční je ilustrován na obrázku **1.4**.



Obrázek 1.4 Převodní systém srdeční (upraveno z [6])

1.2 Elektrokardiografie (EKG)

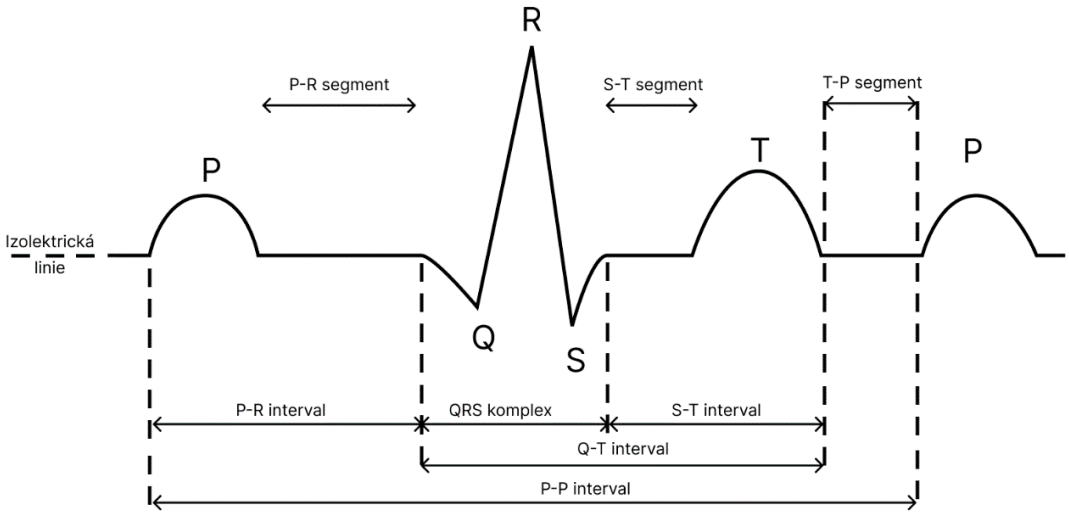
V jednotlivých buňkách pracovního myokardu vzniká v průběhu akčního potenciálu, kde jedna část buňky je depolarizována a jiná část ne, elementární elektrické pole. Buňku lze také tím pádem brát jako dipól, který pak určuje orientaci tohoto elektrického pole. Sumací těchto elementárních elektrických polí v každém časovém okamžiku vzniká okamžitý vektor elektrického pole srdečního. Ten určuje velikost a směr šíření depolarizační (nebo repolarizační) vlny. Tento vektor lze měřit pomocí elektrokardiografických svodů, přičemž naměřená amplituda závisí nejen na velikosti okamžitého vektoru, ale i na jeho orientaci vůči svodu/svodům. [6]

Elektrokardiografie je jedna z nejdůležitějších vyšetřovacích metod srdce. Na základě ní je možné pozorovat fyziologické i patologické stavy srdce. Jak již bylo naznačeno, jedná se o metodu, která je založena na měření elektrické aktivity srdce, respektive měří změny okamžitého vektoru elektrického pole srdečního v čase, která je vyobrazena na signálu zvaném elektrokardiogram v případě povrchového a elektrogram v případě invazivního měření. Díky tomu, že lidské tělo je vodivé, lze signál měřit na povrchu těla, pak se jedná o povrchové EKG, které je neinvazivní. V některých případech je nicméně nutné provést měření invazivně, kde je využito intrakardiálního EKG. [6][9][10] Intrakardiální elektrokardiografie bude v rámci této práce probrána více později.

Měření elektrické aktivity se provádí bipolárně nebo unipolárně. [10] Ty se dále mírně liší dle invazivního a neinvazivního přístupu EKG. V případě neinvazivního EKG se u bipolárního zapojení měří rozdíl mezi dvěma aktivními elektrodami. [6] U invazivního měření rozdíl potenciálů mezi elektrodami na katetru. Unipolární zapojení je měření podobné, v obou případech se jedná o měření potenciálů mezi aktivní elektrodou a indiferentní elektrodou. V případě intrakardiálního EKG je indiferentní elektroda mimo srdce. [10] Směřuje-li u unipolárního depolarizační vlna k elektrodě, je zaznamenána kladná výchylka, od elektrody naopak záporná. U repolarizace platí to stejné, jen obráceně. Podobné platí pro bipolární svody, tedy, směřuje-li průmět depolarizační vlny ke kladné elektrodě, je zaznamenána na svodu kladná výchylka a naopak. U repolarizace platí analogicky to samé, ale naopak. Směřuje-li vektor kolmo, není u bipolárních svodů průmět žádný a tím pádem není zaznamenána žádná výchylka křivky. [6]

1.2.1 EKG signál

Na fyziologickém signálu EKG pozorujeme vlny, komplexy, intervaly a úseky, přičemž úsek je nazývána část signálu mezi dvěma vlnami nebo komplexy. Zejména u povrchového EKG je pozorován charakteristický signál (obrázek **1.5**), který je znám i mezi laickou veřejností, kde je za fyziologických podmínek přítomna vlna P, QRS komplex, vlna T a někdy i vlna U. [10] Jelikož se pro zapojení povrchového EKG používá jisté konvence zapojení (zmíněno níže), bude EKG signál popsán na signálu povrchového EKG.



Obrázek 1.5 Signál povrchového EKG (převzato a upraveno z [11])

Depolarizaci a tím pádem systolu síní za fyziologických podmínek zajištuje SA uzel, jejíž vlna se šíří dále předsíní. Jelikož jsou předsíně tenkostěnné, je zaznamenána malá amplituda. Na EKG signálu se tato událost projeví jako vlna P. Vlna pokračuje k AV

uzlu, kde dojde ke zbrzdění přesunu ze síní do komor, které je pozorování za ideálních podmínek jako izoelektrická linie, označována jako PQ úsek. Následuje depolarizace komor a je manifestována QRS komplexem, kdy na vrcholu R dochází v srdci k izovolumetrické kontrakci. Jelikož je svalovina komor mohutnější oproti síním je výchylka QRS vyšší než u vlny P. Po depolarizaci síní je po chvíli nulová elektrická aktivita, dochází k takzvané plató fázi a na EKG signálu je zaznamenán jako úsek ST. Po klidové aktivitě nastává repolarizace (a tím pádem i diastola) komor. Může se vyskytovat i vlna U, jejíž původ je nejasný. Může se jednat o repolarizaci Purkyňových vláken. [6], [11]

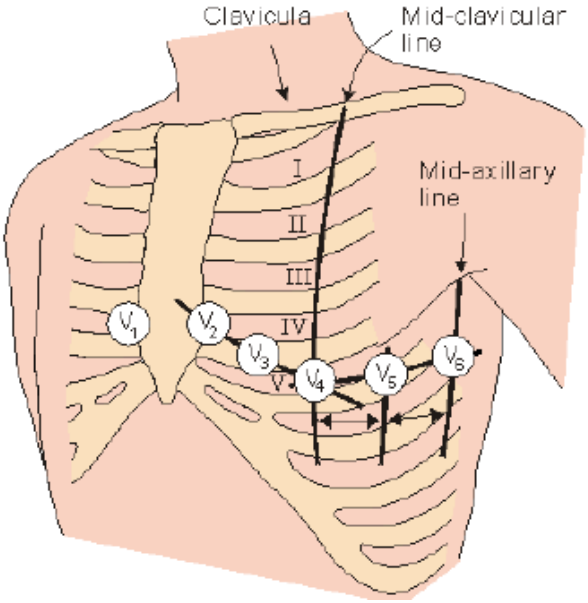
1.2.2 Povrchová EKG

Snímání povrchového EKG se pevně ustálilo na 12 svodovém EKG [12], které lze rozdělit do 3 skupin: bipolární (Einthovenovy) končetinové svody, unipolární (Goldbergovy) svody a unipolární hrudní svody. Končetinové svody, jelikož jsou bipolární, snímají signál mezi dvěma aktivními elektrodami. Goldbergovy svody a hrudní svody jsou naopak unipolární a signál je tak snímán mezi zkoumanou oblastí elektrody a indifreretní elektrodou. [6][10]

Bipolárních končetinové svody se označují římskými číslicemi, I, II a III a také jsou dle polarity označeny znaménkem + nebo - (kladně polarizována elektroda zaznamenává kladnou hodnotu, je-li rozdíl potenciálů kladný a záporně polarizované elektrody je naopak v tomto případě registrována záporná hodnota. Je-li rozdíl potenciálů záporný, nastává analogicky to samé, jen naopak). Elektrody jsou umístněné na pravé i levé horní končetině a levé dolní končetině, tvořící takzvaný Einthovenův trojúhelník. [6]

Unipolární, Goldbergovy, svody jsou označeny jako aVL, aVF a aVR. Jedná se o upravené unipolární zapojení podle Wilsona, kde indiferentní elektroda byla tvořena spojením všech tří končetinových svodů přes odpor 5 KΩ do centrální, Wilsonovy, svorky. V případě zapojení podle Goldberga je z Wilsonovy svorky odpojena jedna končetina, která je snímána proti zbylým dvěma spojeným svodům. Centrální svorka tak již nemá nulový potenciál a výchylka signálu je tak díky tomuto zapojení zvýšena. [6] Proto se těmto svodům také říká jako augmentované svody. [7]

Unipolární hrudní svody, někdy také nazývány jako prekordiální, sledují elektrickou aktivitu v horizontální rovině (zatímco předchozí zmíněné v rovině frontální), čímž lze získat prostorový dojem o rozložení elektrického pole srdce. Jsou označovány V1 až V6, přičemž měřený, aktivní, svod je měřen vůči Wilsonově svorce. Umístnění elektrod je ilustrován na obrázku **1.6**. [6][9][12]



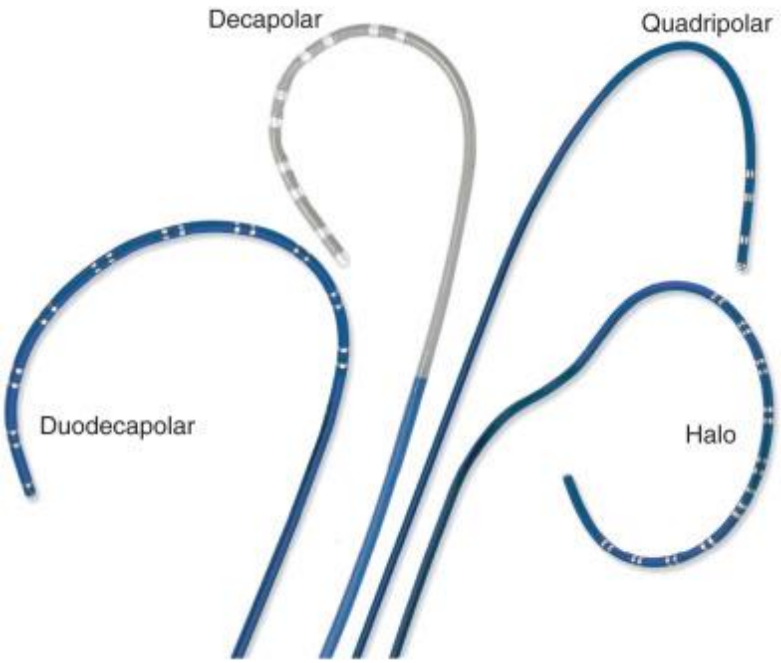
Obrázek 1.6 Unipolární hrudní svody [12]

Povrchové EKG je díky neinvazivnímu přístupu velice používané. Nicméně, jsou případy, kdy použití povrchového EKG není dostačující a je lepší přistoupit k invazivnímu zákroku. Například když je nutné přesně namapovat arytmie, například kvůli radiofrekvenční ablaci [9][10] Další možné využití invazivního zákroku je při zavádění kardioverter-stimulátoru pro snadnější nalezení depolarizačního prahu [13]. V obou případech může být využito intrakardiálního EKG, kterému se věnuje tato práce.

1.2.3 Intrakardiální EKG

Hlavním rozdílem oproti povrchovému EKG je skutečnost, že se jedná o invazivní přístup, kdy se elektrody nachází na diagnostických katetrech, které jsou zavedeny do těla pacienta, obvykle za skiaskopické kontroly. Katetry (obrázek **1.7**) lze dle volitelnosti zakřivení rozdělit na neřiditelné, kdy zakřivení je fixní a řiditelné, u kterých lze zakřivení nastavit. Dle počtu párů elektrod lze zmínit 2-, 4-, 6-, 8-, 10- nebo 20- polární katétry. Nejčastěji se však využívají 4-polární, kvadrupolární, katétry se dvěma páry elektrod, kdy dochází současně na proximálním páru k zaznamenávání signálu a na distálním ke stimulaci.

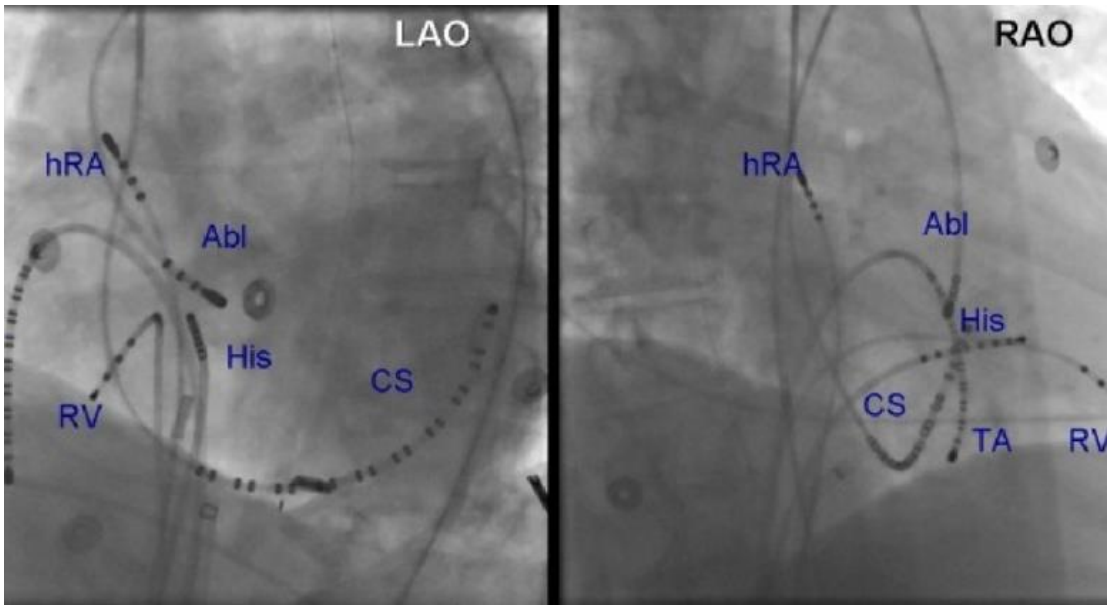
Délka jednotlivých elektrod se pohybuje kolem 1 až 2 mm. U katetrů se rovněž zohledňuje i vzdálenost mezi elektrodami, která bývá 1 až 10 mm (může být i více). Různé vzdálenosti jsou vhodné pro jiný účel. Krátká vzdálenost mezi elektrodami se hodí pro přesné snímání malé oblasti, zatímco velké vzdálenosti elektrod například pro posouzení směru šíření vzruchu. [10][14]



Obrázek 1.7 Multipolární katetry s různým počtem elektrod a tvarů [14]

Zatímco se zapojení, tedy i umístění, svodů u povrchového EKG pevně ustálilo u intrakardiálního EKG nikoliv. Sice jsou dány oblasti pro umístění, jako oblast koronárního sinu (CS), Hisova svazku (His) nebo pravé síně (RA), ale konkrétní přístup umístění katetru v těchto oblastech není přesně dán. Například v pravé síni se katetr většinou umisťuje v horní části laterární stěny poblíž sinoatriálního uzlu nebo v oušku pravé síně [10]

Umístění katetru je prováděno obvykle za skiaskopické kontroly (obrázek **1.8**) a její správnost až po určení ostrosti záznamu dle umístění katetru. Je většinou využito předozadní, pravé šikmé (RAO) a levé šikmé projekce (LAO). Pro umístění katetru v oblasti koronárního sinu je výhodná levá šikmá projekce, pro oblasti Hisova svazku a pravé síně předozadní. Pravou šikmou lze využít pro umístnění oblasti hrotu pravé komory a již zmíněného Hisova svazku. [10][15]



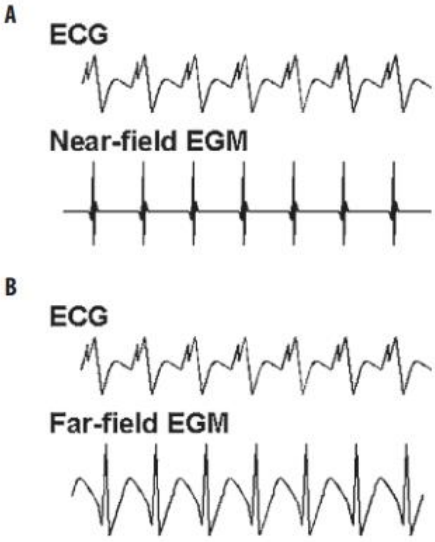
Obrázek 1.8 Ukázka levé šikmé (LAO) a pravé šikmé (RAO) projekce [15]

Oproti povrchovému EKG se také liší i morfologie snímaného signálu. Signál povrchového EKG je směrový, tedy, pokud depolarizační vlna směřuje k elektrodě, zapisuje se kladná výchylka a pokud směřuje od elektrody, tak naopak záporná. U intrakardiálního EKG je signál nicméně lokální záznam aktivity z bezprostřední blízkosti elektrody katetru. Tento signál je ostrý a označuje se jako "near field". Pokud je zaznamenána aktivita ze vzdálenějšího místa na tom samém katetru, je signál méně ostrý a je nazýván jako "far field". [9][10]. "Near field" a "far field" (obrázek **1.9**) se liší i svojí morfologií, kdy "near field" je morfologicky užší a není zachycena síňová aktivita. "Far field" je naopak širší a komorová aktivita je viditelná. [16] Ten se může překrývat s "near field", proto je nutné volit přístupy, díky kterým se oba signály oddělily. Lze využít stimulace ze vzdálenějších míst, čímž se signály časově oddělí. [9]

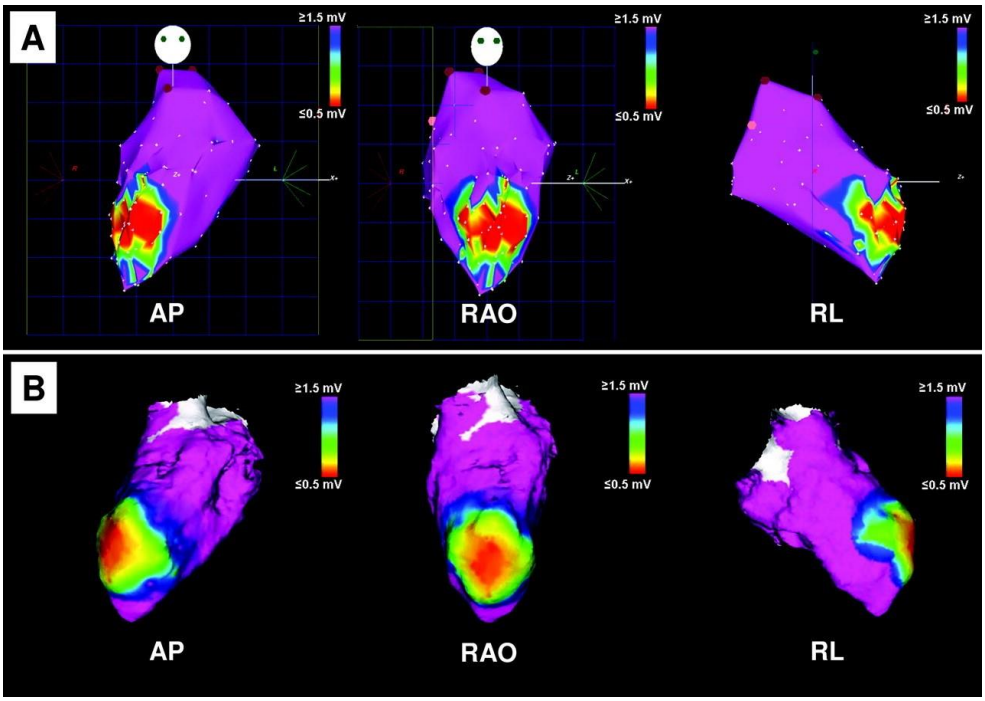
Při patologických stavech lze sledovat například tvz. dvojitý potenciál. Vzniká, když signál přichází ze dvou různých míst, což se pak projeví dvěma charakteristickými výchylkami, při jednom srdečním úderu, oddělených izoleketrickou linií. Bývá zaznamenán v místech lokálního bloku šíření depolarizační vlny. Bývají jako ukazatel úplného bloku trikuspidalní šíje během probíhající ablace u flutteru síní. Jindy lze pozorovat i frakcionovaný elektrogram, což je signál s mnohačetnými komponentami. Tato místa lze zaznamenat v oblastech pomalého vedení, což mohou být místa, kde jsou jednotlivá svalová vlákna oddělena fibrotickými překážkami (například jizvy po infarktu myokardu). Frakcionovaný elektrokram může sloužit pro cílení ablace fibrilace síní. [9], [10]

Spojením informace lokalizace katetru s intrakardiálním EKG lze docílit elektroanatomického mapování, čímž lze získat 3D geometrii srdce s barevnou mapou

reprezentující data elektrokardiogramu, jako například napětí (napěťové mapování, obrázek **1.10**), nebo při porovnání načasování záznamů elektrické aktivity s povrchovým EKG zjistit i informace o propagaci vzruchu (aktivační mapování). [17][18][19] Mezi nejčastěji používanými systémy jsou například CARTO (Biosense Webster, Inc.), EnSite NavX a EnSite Velocity (St. Jude Medical). [10]



Obrázek 1.9 "Far field" a "near field" elektrogramy [16]



Obrázek 1.10 3D napěťové mapování za pomoci systémem CARTO za skiaskopické (A) a MRI (B) kontroly, Zkratky zobrazení: AP – předozadní, RAO - pravá šikmá, RL - pravé boční [18]

1.3 Číslicové zpracování signálů

Naměřené signály obsahují většinou, kromě užitečné informace, také řadu různých rušivých elementů, které mohou následnou analýzu a klasifikaci signálu ovlivnit. Tyto rušivé složky mohou být způsobeny například rušením z elektrické sítě (síťový brum), což je téměř harmonické rušení na 50 Hz, myopotenciály, což je vysokofrekvenční šum, vzniklý činností svalstva pacienta. Zde se může jednat o pásmo i 100 Hz a výše. Dále se často vyskytují i respirační artefakty, což je nízkofrekvenční šum, který se může pohybovat kolem 0,5 Hz. [10][20]

Nejčastěji je využito filtrace, jejíž úkolem je potlačit rušivou složku a zároveň zachovat co nejvíce užitečné informace. Pro síťový brum je využito úzkopásmové zádrže, která slouží k potlačení 50 Hz (někde 60 Hz). Pro vysokofrekvenční šum využito dolní propusti, která obvykle potlačuje pásmo od 300-500 Hz. V případě respiračních artefaktů je využito horní propustí, kdy v případě bipolárních signálů potlačuje do 30 Hz a u unipolárních již od 0,5 Hz. [10][20]

1.3.1 Konverze vzorkovací frekvence

Někdy může nastat případ, kdy je nutné změnit vzorkovací kmitočet. Například u vlnkové transformace nebo když jsou dva systémy, kdy každý pracuje s rozdílnou vzorkovací frekvencí. Například EKG systém může pracovat s jinou vzorkovací frekvencí než software, který provádí zpracování signálu. V takovém případě je vhodné provést konverzi vzorkovací frekvence signálu z EKG systému pro potřeby software, který signál následně zpracuje. [21][22]

V případě zvýšení vzorkovací frekvence se jedná o tvz. interpolaci, kde se jedná o zvýšení vzorkovacího kmitočtu celočíselným faktorem L. To je provedeno tak, že je vloženo $L-1$ nulových vzorků mezi sousední vzorky signálu a následně je provedena filtrace interpolačním filterm (dolní propust). [22]

Snížení vzorkovacího kmitočtu, decimace, je využito celočíselného faktoru D. Nejdříve se aplikuje decimační filtr (dolní propust), aby se zabránilo aliasingu. Následně se vybere D-tý vzorek signálu. [22]

1.4 Zpracování obrazů

Ačkoli je v rámci této práce využito 1D intrakardiálních záznamů, je pro realizaci modelu využito technik ze zpracování obrazů. V této sekci tak bude popsána ve stručnosti problematika segmentace, konkrétně pak prosté prahování, nalezení prahu pomocí metody Otsu, pseudobarvení a morfologické operace, eroze a dilatace.

1.4.1 Segmentace

Segmentaci lze definovat jako rozdělení obrazu na homogenní oblasti dle nějakého kritéria (například dle jasu). Tyto oblasti by měly ideálně korespondovat s konkrétním

objektem scény a neměly se překrývat. Výstupem segmentace může být binární obraz, pakliže se hledá jedna oblast. Pixely této oblasti nabývají hodnoty 1 (nebo 255, záleží, jakého datového typu je obraz) a zbytek 0 (nebo naopak). Je-li hledaných oblastí více, může být využito barevného odlišení (pomocí pseudobarvení, které je popsáno níže). [24] [25]

Segmentace může být částečná (oblasti nekorespondují s objekty vstupního obrazu), kompletní (oblasti souhlasí s objekty vstupního obrazu) [26], **sémantická** nebo **instanční**. Pro potřeby této práce a kvůli obsahové návaznosti dalších kapitol budou popsány pouze sémantická a instanční segmentace. Sémantickou segmentace lze chápat jako rozdělení tříd, nikoliv jednotlivých objektů. Například při sémantické segmentaci modelů aut, na obrázku **1.11**, je pozadí černé barvy a pixely spadající do třídy auto barvy bílé. [27]

Obrázek 1.11 Ilustrovaná sémantická segmentace. Vpravo vstupní obrázek, vlevo výstup segmentace

Naproti tomu instanční segmentace (obrázek **1.12**) provádí rozdělení objektů spadajících do nějaké třídy. Při použití stejného příkladu jsou autíčka také rozděleny od pozadí, ale navíc jsou zohledněny i jednotlivé objekty, tudíž modely aut jsou již samostatné segmenty. [27]

Obrázek 1.12 Instanční segmentace. Vpravo vstupní obrázek, vlevo výstup segmentace, kde je třída auto oddělena od pozadí, nicméně jsou od sebe rozlišeny i jednotlivá auta jako samostatné entity

Technik pro segmentaci je celá řada, jednou z nich, a nejjednodušší, je **prosté prahování**. Principem této techniky je porovnání například hodnoty jasu s předem určenou prahovou hodnotou. Na základě skutečnosti, zda-li je hodnota vyšší či nižší než práh, je tomuto pixelu přidělena předem stanovená hodnota. Pokud je obraz f(x,y) šedotonóvy a práh T, výstupem je binární obraz g(x,y), kde pixelům, jejíž hodnoty jasu jsou vyšší nebo rovny než práh, je přidělená barva bílá, jinak černá. Zmíněné lze zapsat dle matematické formulace (1.1),

$$g(x,y) = \begin{cases} 1 & když\ f(x,y) \geq T \\ 0 & v\ jiném\ případě \end{cases}. \text{[28]} \tag{1.1}$$

Problémem této metody je neefektivní segmentace objektů nasvětlených ze strany a správné určení prahu, aby se intervaly hodnot obou tříd co možno nejmenší měrou nepřekrývaly. Je možné jej například odhadnout, určit z histogramu, nebo využít metod pro automatické určení jako například **Otsu**. [25]

Tvůrcem této metody je Nobuyiki **Otsu** a jejím cílem je nalezení takového prahu, aby překryv intervalů hodnot byl co nejmenší. Toho je docíleno iterativně minimalizováním váhované sumy rozptylů dvou tříd, pozadí a popředí. [29], [30] Ta je popsána rovnicí (1.**2**) (ta je dále pak popsána rovnicemi (1.**3**) až (1.**8**)),

$$\sigma_w^2(t) = w_b(t) * \sigma_b^2(t) + w_f(t) * \sigma_f^2(t), , \tag{1.2}$$

$$w_b\,(t) = \sum_{i=1}^{t} P(i)\ , \tag{1.3}$$

$$w_f\,(t) = \sum_{i=l+1}^{l} P(i)\ , \tag{1.4}$$

$$\mu_b\,(t) = \frac{\sum_{i=1}^{t} i*P(i)}{w_b(t)}, \tag{1.5}$$

$$\mu_f\,(t) = \frac{\sum_{i=l+1}^{l} i*P(i)}{w_f(t)}, \tag{1.6}$$

$$\sigma_b^2(t) = \frac{\sum_{i=1}^{t}\left(i-\mu_b\,(t)\right)^2 *P(i)}{w_b(t)}, \tag{1.7}$$

$$\sigma_f^2(t) = \frac{\sum_{i=l+1}^{l}\left(i-\mu_f\,(t)\right)^2 *P(i)}{w_f(t)}, \tag{1.8}$$

kde σ_w^2 je váhovaný součet rozptylů, σ_b^2 je rozptyl třídy pozadí a σ_f^2 je rozptyl popředí. Dále pak $w_b(t)$ a $w_f(t)$ jsou váhy, definovány jako pravděpodobnost pixelu výskytu v dané oblasti. $\mu_f\,(t)$ a $\mu_b\,(t)$ jsou průměry jednotlivých tříd. [30][31] Prahování pomocí metody Otsu je na obrázku **1.13**.

Obrázek 1.13 Prahování pomocí metody Otsu. Na obrázku vlevo si lze všimnout úskalí prahování, tedy nasvětlení objektů ze stran, kdy odlesky kapoty a skla se projevily i u nesprávné segmentace

1.4.2 Pseudobarvení

Cílem této metody je z šedotónového nebo binárního obrazu udělat obraz barevný a to přiřazením barev na základě jejich hodnoty. Barvy neobsahují žádný informativní obsah o skutečných barvách scény. Metoda slouží pouze pro lepší uživatelský vjem, jelikož zdravé lidské oko dokáže lépe vnímat barvy než odstíny šedi. [25][32]

1.4.3 Morfologické operace

Morfologické operace jsou techniky zpracování obrazu, zabývající se morfologií obrazu. Morfologií je myšlen popis tvarů a struktur v obrazu. [33] Bývají používány při zpracování binárních obrazů, ale je možné je použít i pro obrazy šedotónové. Cílem těchto operací je odstranění nedostatků obrazu, například po segmentaci obrazu, jako například dodatečné úpravy struktur obrazu nebo odstranění šumu. Například u binárních obrazů je lze použít například k vyplnění děr, rozšíření, zúžení nebo odstranění některých struktur, které ve výstupním obraze nejsou žádané nebo mohou ovlivnit například tvary struktur v obraze. Je založen na teorii množin, kdy pro je zde jako množina a jsou využívány množinové operace jako například průnik či sjednocení množin, zde obrazů. Hlavními morfologickými operacemi jsou eroze a dilatace. Jiné operace vznikají právě jejich kombinací (otevření a uzavření). [25][33]

Operace využívají tzv. strukturního elementu (obrázek **1.14**), což je struktura podobná maskám jako u konvoluce. Jedná se malou binární matici, která se pohybuje po obraze. Jeho tvar a velikost ovlivňuje podobu výstupního obrazu. [25]

1	1	1
1	1	1
1	1	1

0	1	0
1	1	1
0	1	0

Obrázek 1.14 Ilustrace možných podob strukturních elementů, kde červená hodnota 1 je referenčním bodem (upraveno z [33])

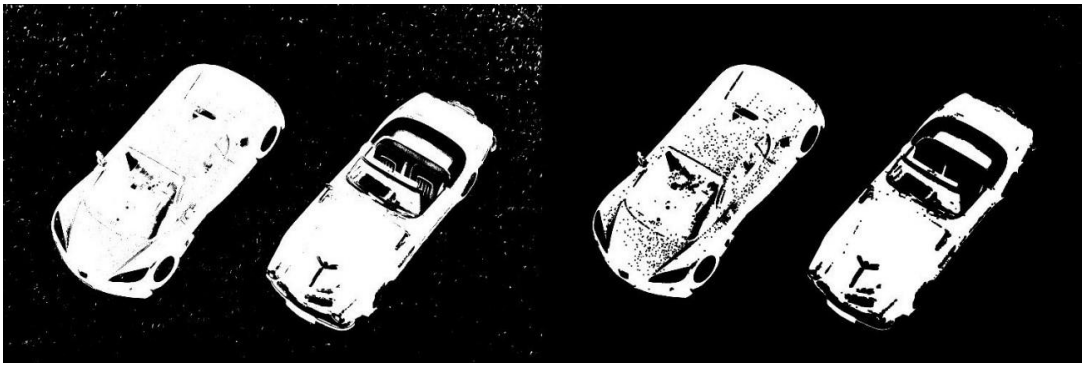
Na určitém místě masky (kdekoliv) se nachází takzvaný referenční bod, na který se ukládá výstup. [25] Ačkoli je v rámci této práce použita jen dilatace, bude popsána i eroze. Jak již bylo zmíněno, jsou klíčovými morfologickými operacemi.

Eroze je morfologickou operací, která zmenšuje objekty. Lze ji tak využít pro zúžení struktur či k odstranění malých objektů, které mohly po prahování vzniknout přítomným šumem v obrazu. Aplikací eroze na vstupní obraz A za využití strukturního elementu B vzniká výstupní obraz C. Jsou-li objekty definované hodnotou jedna a pozadí hodnotou nula, pak u eroze platí, že jsou-li pod všemi jedničkovými hodnotami strukturního B elementu posunutého na pozici x vstupního obrazu A prvky obrazu o hodnotě jedna, je na výstupním obraze C na pozici, kde se nachází referenční bod, napsána hodnota jedna. Erodovaný obraz C vzniklý erozí obrazu A za využití strukturního elementu B je definován rovnicí (1.**9**),

$$C = \{x \mid (B)_x \subseteq A\}, \tag{1.9}$$

kde $(B)_x$ je strukturní element posunutý na pozici x, A je již zmíněný, vstupní obraz a C je obraz výstupní. Vhodnou volbou velikostí a tvaru strukturního elementu je tak dosaženo zmenšení struktur nebo odstranění šumu. [25][33]

Na obrázku **1.15** je ilustrována eroze, kde právě malé struktury, které byly ve vstupním obraze, po aplikaci této operace se ve výstupním obraze již nenachází. Také došlo k ovlivnění struktur, kdy například u autíčka vlevo jsou jednotlivé černé „ostrůvky“ mnohem větší než na vstupním obrazu.

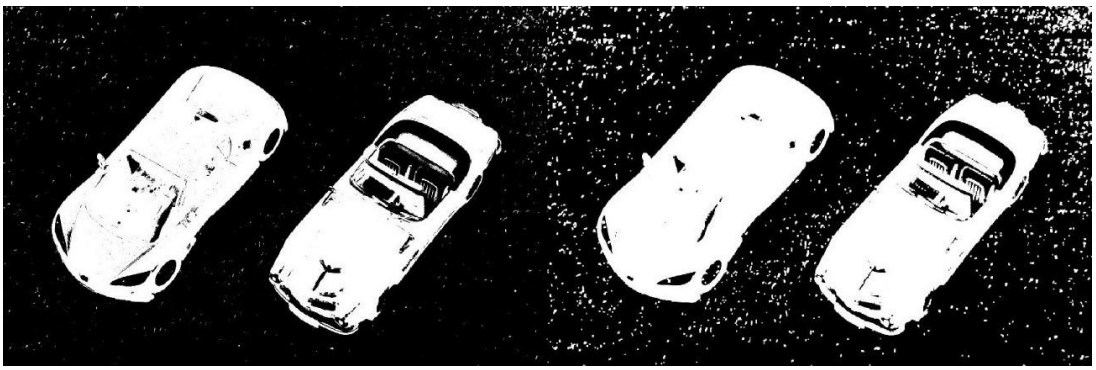


Obrázek 1.15 Erodovaný obraz s použitím strukturního elementu čtverce o rozměrech 11x11

Dilatace lze považovat za komplementární operací k erozi. Není ale reversibilní, to znamená, že pokud je na obraz aplikována eroze a následně dilatace, tak není získán původní obraz. Dilatace, na rozdíl od eroze, objekty zvětšuje a vyplňuje. Zatímco bylo u eroze po připsání hodnoty jedna nutné, aby všechny jedničkové prvky byly pod všemi jedničkovými body u strukturního elementu, u dilatace je to jiné. Je-li alespoň pod jednou jedničkou strukturního elementu prvek obrazu o hodnotě jedna, pak je na výstupním obraze C na pozici, kde se nachází referenční bod, hodnota jedna. Matematická formulace dilatovaného obrazu C je popsána rovnicí (1.**10**),

$$C = \{x \mid (B)_x \subseteq A\}, \tag{1.10}$$

kde $(B)_x$ je strukturní element posunutý na pozici x, A je vstupní obraz [25][33]. Na obrázku **1.16** je zobrazen dilatovaný obraz, u kterého si lze všimnout, že se kromě vyplnění menších otvorů u autíček zvětšily i malé struktury, které zde mají roli šumu.



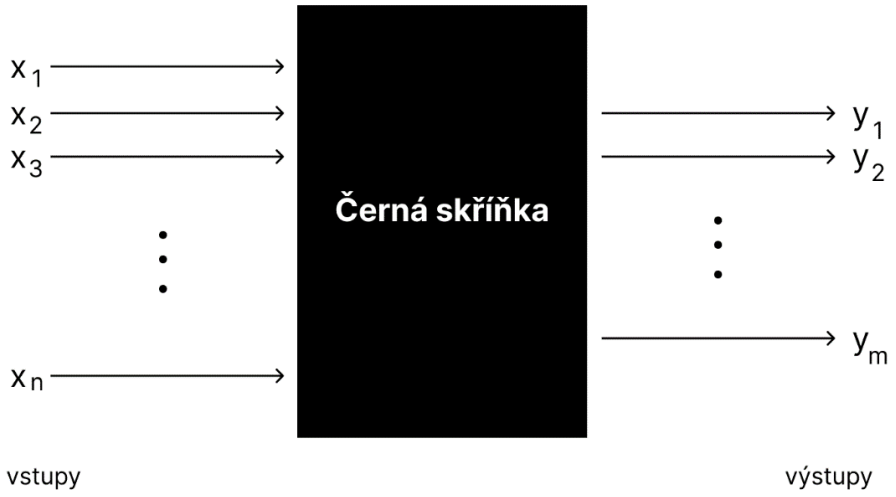
Obrázek 1.16 Původní binární obraz, (vpravo) Dilatace binárního obrazu s použitím strukturního elementu čtverce o rozměrech 11x11

1.5 Umělé neuronové sítě

Proces analýzy a klasifikace signálů byl dříve čistě v rukách lékařů. S rozvojem číslicového zpracování signálů a zejména pak umělé inteligence lze tyto procesy čím dál tím více přenechat strojům, zejména pak pomocí umělých neuronových sítí, které nabírají na stále větší popularitě, v neposlední řadě pak konvoluční neuronové sítě. [34][35][36]

Dříve však, než bude konvoluční neuronová síť popsána, bude z důvodu návaznosti uvedena obecná problematika umělých neuronových sítí. Pro porovnání bude také ve stručnosti zmíněn i vícevrstvý perceptron.

Umělou neuronovou síť lze chápat jako černou skříňku (obrázek **1.17**), která má několik vstupů a výstupů, pracující s několika paralelně zapojenými aritmetickými jednotkami. [37]

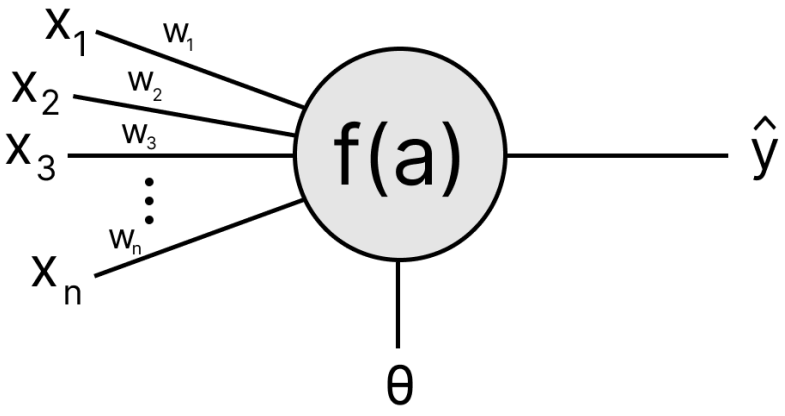


Obrázek 1.17 Umělá neuronová síť jako černá skříňka (upraveno z [37])

Existuje několik typů umělých neuronových sítí. Lze je dle učení zařadit na učení s učitelem (modelu jsou poskytnuty vstupy a příslušné výstupy, na základě kterých se model „naučí“ – úpravou parametrů sítě) a na učení bez učitele, které v rámci této práce nebude věnována pozornost (kdy nejsou modelu poskytnuty před připravené požadované vstupy a výstupy. [37]

Základní jednotkou umělé neuronové sítě je umělý neuron. Síť tvořená jedním neuronem je takzvaný perceptron (obrázek **1.18**).

Do perceptronu vstupuje vektor vstupů, $X = [x_1, x_2, x_3, \dots, x_n]$, například vzorky signálu nebo příznaky. Tyto vstupy jsou násobeny příslušnými vahami $W = [w_1, w_2, w_3, \dots, w_n]$. Následuje suma součinu vstupů s vahami. [37][38]



Obrázek 1.18 Perceptron se vstupy x s příslušnými vahami w, aktivací a, prahem θ a výstupem $\hat{y}$ (upraveno z [38])

Dalším vstupem je práh θ, který se od váhované sumy odečte. Tomuto výstupu se pak říká aktivace a, která vstupuje do aktivační funkce f, jejíž účel a tipy budou zmíněny dále. Matematické vyjádření lze zapsat jako je v rovnici (1.11),

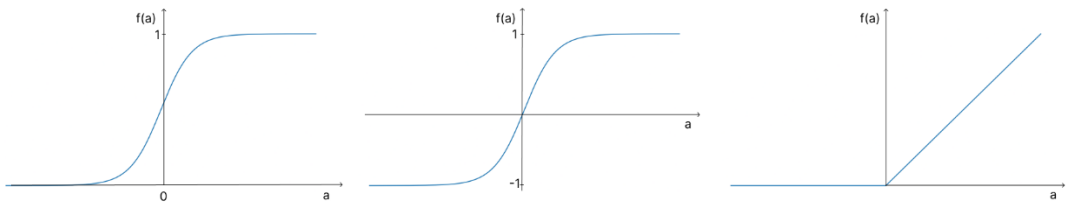
$$f(\sum_{i=1}^{N} x_i w_i - \theta) = f(a) \text{ . [38][39]} \tag{1.11}$$

Stanoví-li se, pro snadnější výpočty, že práh bude roven součinu $x_0 = 1$ s nulovou vahou w0 = $-\theta$, lze pak matematické vyjádření z (1.11) přepsat na

$$f(\sum_{i=0}^{N} x_i w_i) = f(a). \text{ [38][39]} \tag{1.12}$$

1.5.1 Aktivační funkce

Aktivační funkce převádí vstupní data na výstupní. Existuje celá řada aktivačních funkcí, nicméně pro potřeby učení musí by měly být derivovatelné. [40] Níže budou uvedeny pouze derivovatelné funkce. Na obrázku **1.19** jsou ilustrovány některé z nich.



Obrázek 1.19 Aktivační funkce, z leva doprava: Sigmoida, Hyperbolická tangenda, ReLU (upraveno z [38])

Sigmoida je funkce, která nabývá hodnot od 0 do 1. Jedná se o nelineární aktivační funkci, což je žádané, jelikož síť je tak schopná zvládat komplexnější úkoly, jako například přizpůsobení se datům nebo lepší proces učení (zmíněno dále). Její nevýhodou nicméně je, že výstup je vždy nezáporný a nevyužívá celý rozsah hodnot. Pokud by byl vstup, tedy aktivace, příliš velký, výstupem bude stále hodnota 1. To může způsobit

potíže v průběhu zpětného šíření chyby (zmíněno dále), kdy derivace sigmoidy nabývá malých hodnot, blízkých nule, a tak může síť předčasně ukončit učení. Jedná se o takzvaný problém mizejícího gradientu (z angl. vanishing gradients problem). [38][40][41][42][43][40] Je-li aktivace a, je aktivační funkce sigmoida matematicky zapsána rovnicí (1.13),

$$f(a) = \frac{1}{1+e^{-a}}. \text{ [38][39][42]} \tag{1.13}$$

Hyperbolická tangenta (rovnice (1.14)) je podobná funkci sigmoidy, prochází ale počátkem a její rozsah hodnot je -1 a 1, čímže se vyřešil jeden z jejich nedostatků sigmoidy. Problém mizejícího gradientu stále přetrvává. [38][39][42]

$$f(a) = \frac{1}{1+e^{-a}} \text{ [38][39][42]} \tag{1.14}$$

Rectified Linear Unit, zkráceně ReLU (rovnice (1.16)), je jedna z nejvíce používaných funkcí. Jedná se o lineární funkci pro kladné aktivace, a naopak pro záporné hodnoty nabývá výstup vždy hodnoty 0. Díky této vlastnosti je funkce stále nelinární, ale v určitých případech lze použít jako funkci lineární. Také řeší problém mizejícího gradientu. Nevýhodou je, že není příliš vhodná jako výstupní funkce poslední vrstvy, například pro klasifikaci. Pro tyto účely se hodí například Softmax. [40][43][44]

$$f(a) = \max(0, a) \text{ [38][42]} \tag{1.16}$$

Softmax (rovnice (1.17)) je oproti ReLU používána jako funkce výstupní vrstvy pro klasifikační problémy. Vrací sice hodnoty od 0 do 1, ale v tomto případě se jedná o převedené vstupy na pravděpodobnost k určité třídě. Lze jej využít pro klasifikaci do více tříd (K). [40][41]

$$f(a_i)\frac{e^{a_i}}{\sum_{j=1}^{K} e^{a_j}} \text{ [38][40][41]} \tag{1.17}$$

1.5.2 Proces učení

V rámci učení s učitelem vyžaduje síť ve fázi učení u trénovacích dat jak vstupy, tak i výstupy. Proces učení probíhá tak, že se z počátku určí váhy náhodně, například mezi hodnotami 0 a 1. Následně proběhne **dopředné šíření**, tedy výpočet výstupu (predikce) $\hat{y}$ a vypočítá se chyba pomocí chybové (někdy také nazývány ztrátové či kriteriální) funkce, čímž se určí, jak moc se predikce, $\hat{y}$, od požadovaného výstupu, y, liší. [38][45][46]

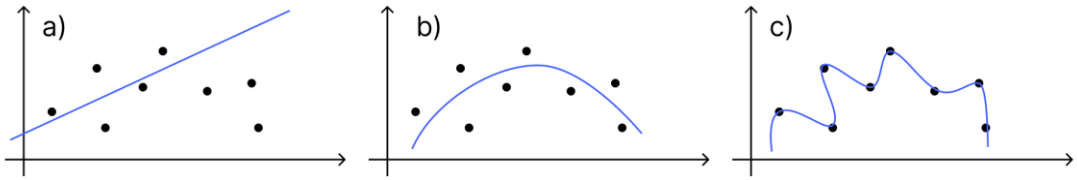
Cílem učení je minimalizace výstupní chyby úpravou vah (a prahů), aby byla chyba co nejmenší. To je dosaženo **zpětným šířením**, což je iterativní metoda úpravy vah na základě chyby z předchozí epochy nebo mini-batch (rozdíly jsou zmíněny v učících algoritmech), založenou na výpočtu derivaci chybové (kriteriální) funkce L, podle dané váhy w_i, tedy $\frac{dL}{dw_i}$ (dále bude uváděna obecněji pro celou síť $\nabla_W L$). Tím je vypočítán

gradient chybové funkce. Tato funkce je nicméně složená. Pro derivaci této funkce lze použít takzvané řetězové pravidlo (1.18), z angl. **chain rule**, díky kterému lze derivaci rozložit jako na součin dílčích derivací, zpětně skrze vrstvy sítě: nedříve výpočet derivace chybové funkce podle predikce $\frac{dL}{d\hat{y}}$, následuje derivace predikce dle aktivace $\frac{d\hat{y}}{da}$ a nakonec derivace aktivace podle dané váhy $\frac{da}{dw_i}$. Díky tomu je výpočet výhodnější. [38] [46][47][48][49]

$$\frac{dL}{dw_i} = \frac{dL}{d\hat{y}}\frac{d\hat{y}}{da}\frac{da}{dw_i}. \text{ [38]} \tag{1.18}$$

Hlavním cílem je dalšími epochami dosáhnout jistého optima. Toho lze docílit nejčastěji hledáním globální minimální chybové funkce pomocí metod **gradientního sestupu**. [48][49] Dříve, než budou popsány jednotlivé metody gradientního sestupu budou zmíněny chybové funkce.

Je nutné volit počet epoch tak, aby nenastalo podučení (underfitting) nebo přeučení (overfitting) a bylo dosaženo optima (obrázek **1.20**). [50]



Obrázek 1.20 Ukázka a) podučení, b) optima, c) přeučení (upraveno z [47], [51])

1.5.3 Chybové funkce

Jak bylo zmíněno v předchozí sekci, pro výpočet chyby, respektive, určení odlišnosti predikce od požadovaného výstupu se vypočítává pomocí chybových (lze také využít termín ztrátových či kriteriálních) funkcí. Existuje několik kriteriálních funkcí, přičemž každá z nich může být vhodná pro jiné využití. Lze je rozdělit v závislosti na účelu učení na regresní a klasifikační. Mezi regresní patří například **MSE** (mean square error), **MAE** (mean absolute error) které nebudou v této práci předmětem zájmu.[38][52] MSE popisuje rovnice (1.19 a MAE (1.20),

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{1.19}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{1.20}$$

kde n vyjadřuje počet epoch. [38] Mezi klasifikační pak patří například **binární křížová entropie**, která je používána pro binární klasifikace. **Entropie** je měřítko míry neurčitosti určité náhodné proměnné. Pokud by byly dvě třídy, například hrušky a jablka byly by pravděpodobnosti příslušnosti k dané třídě náhodné proměnné. Pokud by všechny vzorky byla 100% pravděpodobnost pro třídu hrušky (nebo pouze jablka), není zde

neurčitost. Entropie by byla nulová. Pokud by však byla 50 % pravděpodobnost příslušnosti k hruškám a 50 % k jablkům, tedy, pro obě třídy stejná, je tím pádem vyšší neurčitost. [53] Entropie by byla v tomto případě naopak vysoká. Matematická interpretace binární křížové entropie je formulována rovnicí (1.21),

$$L = \frac{1}{m} \sum_{i=1}^{m} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i), \tag{1.21}$$

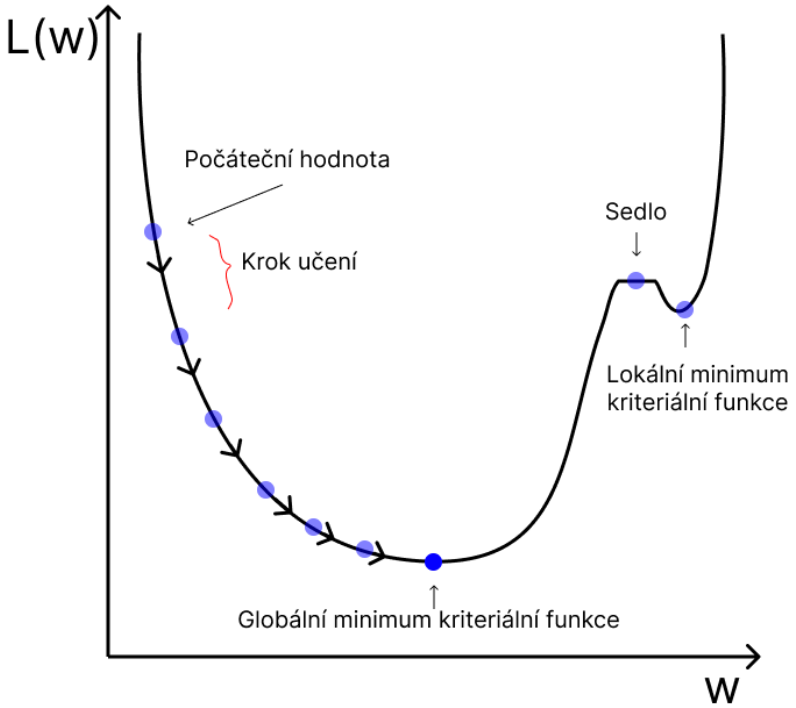
kde m je počtem vzorků. [38], [53] Tato chybová funkce vyžaduje, pro klasifikaci do dvou tříd (například pro segmentaci, kdy 1 je oblastí zájmu a 0 zbytek), aby ve výstupní vrstvě sítě byla aktivační funkce, jejíž výstup bude rozsahu hodnot od 0 do 1. Tou se nabízí sigmoida. Pro klasifikaci do více tříd lze využít **křížovou entropii** (1.22), případně její rozšířenou verzí, váhovanou křížovou entropii (1.23). Kde m tentokrát značí počet tříd a β_c značí váhu dané třídy c,

$$L = \sum_{c=1}^{m} y_c \log(\hat{y}_c), \tag{1.22}$$

$$L = \sum_{c=1}^{m} \beta_c y_c \log(\hat{y}_c). \text{ [38], [52], [54]} \tag{1.23}$$

1.5.4 Učící algoritmy

Učících algoritmů je k dispozici celá řada, nicméně, klíčovou metodou pro nalezení globálního minima chybové funkce, dle které vychází i další metody je Gradient descend, neboli metoda gradientního sestupu.[47][48][49][55]



Obrázek 1.21 Ilustrace metody gradientního sestupu a problémových oblastí

Principem je směřování počátečního bodu, v tomto případě chyby dle váhy *L(w),* proti směru gradientu chybové funkce (obrázek 1.21). S využitím této metody jsou parametry sítě upravovány tak dlouho, dokud nedosáhnou optima. [55]

Úprava parametrů probíhá po epochách, tedy, pro změnu parametru jsou použity všechny vzorky trénovacího souboru dat (někdy se také používá výraz dávka – batch) [47]. Jinými slovy, provede se dopředné šíření všech vzorků, pomocí chybové funkce se určí chyba a teprve až potom se parametry upraví. Matematicky je proces formulován rovnicí (1.24) následovně,

$$W_i = W_{i-1} - \mu \nabla_w L(x, y, W_{i-1}), \tag{1.24}$$

kde *i* vyjadřuje epochu a tím pádem *i-1* epocha předchozí, *x* a *y* reprezentují trénovací soubor dat a μ je krok učení, který je neměnný. [38][55] Úskalím této metody je především vysoká výpočetní náročnost, jelikož je pro změnu parametru využitý celý soubor dat. Dalším nedostatkem je problém se sedly (nulový gradient) nebo lokálními minimy funkce, kvůli kterým nemusí optimalizace pokračovat (obrázek 1.21). [38]

Dříve než budou popsány další metody, bude ve stručnosti zmíněna problematika extrémů funkce. Extrémy funkce jsou maxima a minima, přičemž tyto extrémy mohou být buď globální nebo lokální. S extrémy je spjatý tzv. **stacionární bod** (1.25), což je bod křivky, kde je derivace rovna nule [56][57], tedy

$$f'(x) = 0. \tag{1.25}$$

Stacionární bod je **podezřelý z extrémů**. Zda se v tomto bodě jedná o extrém či nikoliv je možné určit pomocí druhé derivace funkce, $f(x)''$, přičemž, je-li hodnota záporná (a nenulová), pak se jedná o maximum a je-li hodnota nezáporná (a nenulová), jedná se o minimum. [56]

Je-li hodnota z druhé derivace rovna nule, se jedná o tzv. **inflexní bod**, který slouží k určení ke změně konvexnosti a konkávnosti funkce. Je-li bod stacionárním bodem a zároveň bodem inflexním, tedy, první derivace funkce je rovna nule, ale nejedná se o extrém, pak se jedná o takzvané sedlo, respektive, **sedlový bod**. [57][58]

Další metoda, **Stochastický gradientní sestup** (Stochastic gradient descend), popsána rovnicí (1.26), oproti prosté metodě gradientního sestupu, naopak provádí úpravy parametrů každou iteraci. Pro úpravu parametrů sítě je tedy využitý každý vzorek. V rámci epochy je tedy provedeno *N* iterací, kdy *N* je myšlen počet vzorků trénovacího souboru dat. Matematické vyjádření je stejné, jako u předchozí metody, nicméně i není tentokrát myšlena epocha, ale iterace. Metoda je také výpočetně náročná. [38][55]

$$W_i = W_{i-1} - \mu \nabla_{w_i} L(x_i, y_i, W_{i-1}). \tag{1.26}$$

Mini-dávková metoda gradientního sestupu (Mini-batch gradient descend) kombinuje obě předchozí metody, tedy, parametry jsou upraveny až po n iteracích (tzv. batch – dávka). Díky tomu je tato metoda výpočetně přijatelnější. Matematická formulace (1.27) je podobná rovnici (1.26), [38][55]

$$W_i = W_{i-1} - \mu \nabla_{w_i} L(x_i, y_i, W_{i-1}). \tag{1.27}$$

U všech zmíněných metod je hlavním nedostatkem pevný krok učení, kdy jeho určení může být obtížné. Pokud by se zvolil příliš malý krok, učení by trvalo příliš dlouho. Pokud by se zvolil naopak příliš velký krok, mohlo by dojít k „přestřelení“ globálního minima, čímže může dojít i ke zvýšení chyby (namísto zmenšení). Dalším nedostatkem je skutečnost, že stále přetrvává problém s lokálními minimy a sedly chybových funkcí. Problémy s lokálními minimy a sedly lze vyřešit použitím hybnosti. Díky ní je akcelerován sestup gradientů, a tak se v každém kroku nevychází pouze z aktuálně vypočítaného gradientu, ale jsou zohledněny i směry z předchozích kroků. [55]

Hybnosti využívá například upravená Stochastická metoda gradientního sestupu s hybností, která si sice lépe poradí například se sedly, krok učení je nicméně stále neměnný. [38]

Naproti tomu velmi používaný **Adaptivní odhad hybnosti** (Adaptive moment estimaition), zkráceně Adam, disponuje modifikovaným krokem učením a navíc, je využito také exponenciálního zapomínání pro efektivnější úpravu učebního kroku. [37] [38][55] Metoda je popsána rovnicemi (1.28) až (1.30),

$$W_i = W_{i-1} - \frac{\mu}{\sqrt{v_{i-1}+\epsilon}}\, m_{i-1}, \tag{1.28}$$

$$m_i = \beta_1 m_{i-1} - (1-\beta_1)\nabla_{w_{i-1}} L, \tag{1.29}$$

$$v_i = \beta_2 v_{i-1} - (1-\beta_2)\left(\nabla_{w_i} L\right)^2, \tag{1.30}$$

kde m představuje hybnost, v je tzv. root mean square propagation, jedná se o exponenciální váhovaný průměr gradientů z předchozích kroků. β_1a β_2 jsou parametry zapomínání. ϵ je parametr, který slouží jen jako prevence, aby se nedělilo nebo neodmocňovalo nulou, nabývá velmi nízké hodnoty blízké nule. [38][55]

1.5.5 Regularizace

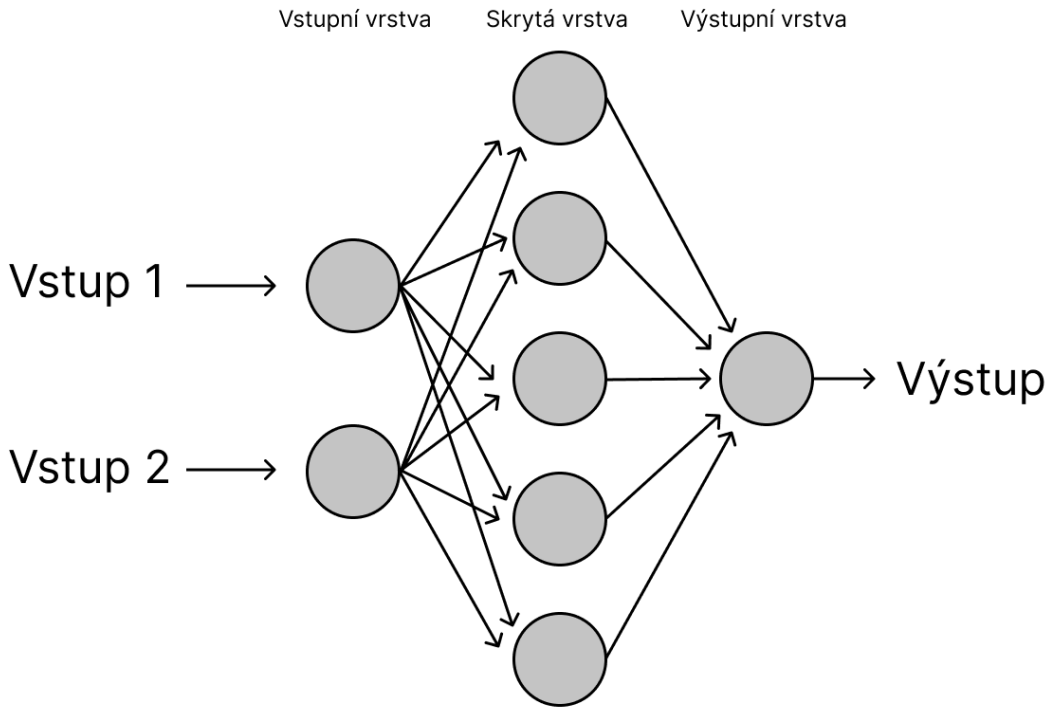
Jako prevence přeučení sítě lze využít metod regularizace. Jednou z nich je takzvaný **DropOut**, což lze chápat jako náhodné vypínání neuronů, dané parametrem p, který udává pravděpodobnost, že neuron bude během trénování zachován nebo vypnut (nezapojí se v kroku učení). [38][50]

Batch normalizace je metoda, která normalizuje průměry a rozptyly mini-batch určitého příznaku před vstupem do další vrstvy. [38]

1.5.6 Vícevrstvý perceptron

V případně využití samotného perceptronu jde pouze o prosté prahování. Jeho hlavní limitací je skutečnost, že je vhodný pouze pro lineárně separabilní problémy.

Tento nedostatek lze vyřešit spojením více perceptronů do známé vícevrstvé neuronové sítě (obrázek **1.22**) nebo také nazývané jako dopředné neuronové sítě. Z hlediska učení se jedná o učení s učitelem. Je tvořen třemi plně propojenými vrstvami, a to vstupní vrstvou, skrytou (může jich být více) a výstupní vrstvou. [41][59][60]



Obrázek 1.22 Vícevrstvý perceptron (upraveno z [60])

1.5.7 Úskalí plně propojených neuronových sítí

Ačkoliv je vícevrstvý perceptron velmi známý, má své nedostatky. Úspěšnost sítě je závislá na příznacích, které jsou jí poskytnuty. Nesplňují-li příznaky požadavky sítě, je síť neefektivní. Tento nedostatek řeší hluboké neuronové sítě, které jsou schopné i extrakce příznaků. [61]

V případě vícevrstvé neuronové sítě ale nastává problém s výpočetní náročností, jelikož je každý neuron propojen s každým neuronem z předchozí vrstvy (tzv. i více parametrů). Tento nedostatek řeší konvoluční neuronová síť. [61][62]

1.6 Konvoluční neuronové sítě

V konvoluční neuronové síti je každý neuron v dané vrstvě spojen jen s některými neurony z předchozí vrstvy, přičemž s nimi sdílí stejné parametry (váhy). Tato vlastnost znamená výrazné snížení výpočetních nároků oproti vícevrstvému perceptronu. [62][63]

Její název plyne z vrstev, které využívají konvoluci. Je známá zejména v rámci zpracování obrazů, má však uplatnění i u 1D signálů. Je tvořena několika vrstvami, mezi

nejčastější však patří konvoluční, pooling vrstvy a plně propojené (fully conected) vrstvy. [62] Dále mohou být přítomny vrstvy nadvzorkování jako unpooling. Ten ale musí být v kombinaci s pooling vrstvou. [64]

1.6.1 Konvoluční vrstva

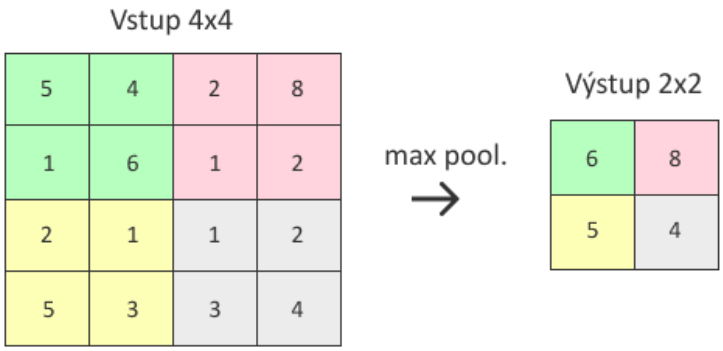
Jedná se o nezbytnou vrstvu konvoluční neuronové sítě. Slouží k extrakci příznaků. V mělčích vrstvách se jedná například o hrany a v hlubších mohou být naopak detekovány části objektů a detailnější geometrické tvary. Neuron je v podobě filtru (lze použít i výraz kernel), jehož aktivace se vypočítá lineární kombinací prvků vstupu s váhami. Aktivace dalšího neuronu v rámci dané vrstvy se provede stejně, jen se okno filtru posune dále. Lze tak celý proces nahradit konvolucí. Výstupem je tzv. mapa příznaků. Je-li v konvoluční vrstvě více filtrů s jinými parametry, lze tak vypočítat více příznakových map. Výstupem z této vrstvy tak může být několik příznakových map poskládaných za sebou do hloubky. [62][63][65]

Výstup závisí na hyperparametrech. V rámci této vrstvy se jedná o počet filtrů, který určuje hloubku výstupu. velikost filtru, padding (pro určení, zda chceme počítat konvoluci z validních nebo i nevalidních vzorků) a krok konvoluce (Stride). [63][65]

1.6.2 Pooling vrstva

Tato vrstva slouží k redukci dimenze vstupu, aby se redukoval počet parametrů, díky čemuž se redukují i výpočetní nároky. Navíc je tato vrstva vhodná pro extrakci dominantních příznaků. Jsou dva typy poolingu a to max a průměrující. [62][63][65]

Proces poolignu (obrázek **1.23**) je jednoduchý, vybere ze vstupu oblast, vymezena pohyblivým oknem o daném rozměru a vybere se buď maximum (max pool) nebo se vypočítá průměr (average pool). I zde jsou parametry jako rozměr, což je oblast, na které se provede pooling. Dále pak stride, což je krok pohyblivého okna. Oba tyto parametry jsou rozhodující pro výsledný rozměr. Například když je stride nízký, je rozměr výstupu vyšší, než kdyby krok byl vyšší. [62][63]



Obrázek 1.23 Ukázka max poolingu. Velikost okna 2x2, stride 2 (upraveno z [64])

1.6.3 Plně propojená vrstva

Jedná se o vrstvu, který bývá většinou před výstupní vrstvou. Zatímco konvoluční a pooling vrstvy lze souhrnně onačit jako extraktory příznaků, plně propojená vrstva slouží ke klasifikaci či regresi. Aktivačními funkcemi může být například ReLU nebo SoftMax. Může být využita i regularizace, jako například dropout.[62][63][65]

1.6.4 Regularizace

Mohou být přítomny i vrstvy jako batch normalizace – normalizuje vstupy do další vrstvy [38]. Dropout vrstva – vypínání neuronů [50]. Obojí se většinou vkládají před konvoluční nebo plně propojenou vrstvu. [63]

1.6.5 Skip connections

Jedná se o propojení, díky kterým lze přeskočit do hlubších vrstev a zachování informace (mapy příznaků), které by se jinak v průběhu průchodu všech vrstev ztratily (je nutné, aby byly stejné prostorové dimenze). S větším počtem vrstev roste i počet dílčích derivací pro výpočet gradientu (řetízkové pravidlo). Jeli těchto dílčích derivací mnoho a mají hodnotu menší než 1 (což většinou mají), je násobením výsledný gradient velmi malý, téměř nulový. Tomuto jevu se říká mizející gradient. To se může projevit pozastavením poklesu trénovací chyby v průběhu učení. [46][43][63][66]

Využitím skip connections lze tak využít informace z předchozích vrstev ve vrstvách hlubších, čímž je využito další informace, a navíc lze eliminovat problém mizejícího gradientu. Jedním z přístupu, který zle k propojení vrstev použít je takzvaná konkatenace, kdy se jednotlivé aktivační mapy dávají za sebe do hloubky, rozměry jsou nicméně zachovány. Další možností je sčítání, kdy se jednotlivé odpovídající pixely sčítají. Na rozdíl od konkatenace nedochází ke změně hloubky. [63][66]

1.6.6 1 x 1 Konvoluční vrstva

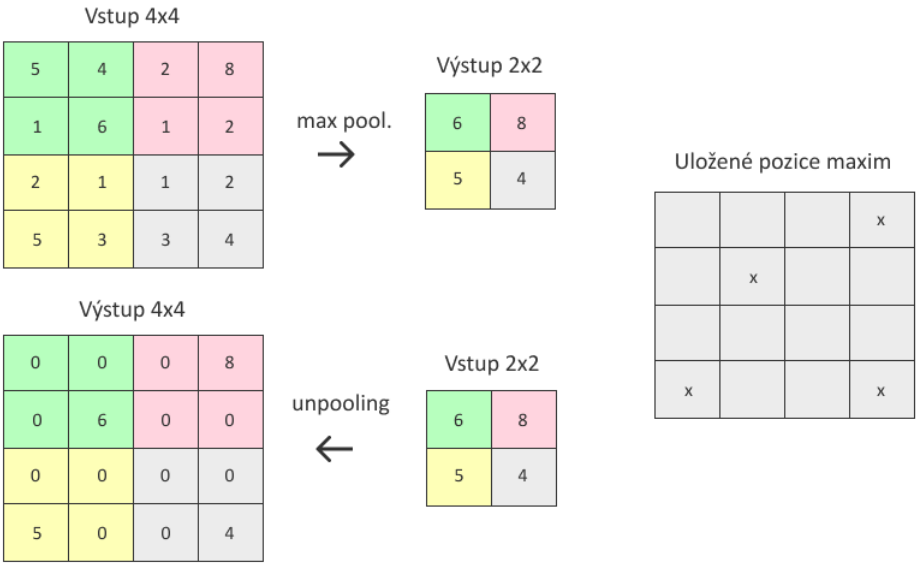
Jedná se o konvoluci s filtrem o velikosti 1x1. Jejím hlavním využitím je změna dimenzionality kombinováním (lineární kombinací) příznaků do hloubky, čímž se sníží výpočetní nároky. Výstup má stejné rozměry, ale počet kanálů je změněn. Lze využít jako alternativu k plně propojené vrstvě. [63][67]

1.6.7 Vrstvy nadvzorkování

Nejčastěji používanou pooling vrstou jsou vstupy podvzorkovány. Vrstvy nadvzorkování jsou užitečné v případě, kdy je potřeba abstraktní reprezentaci vstupu převzorkovat zpět do rozměrů vstupu. Je k dispozici několik vrstev. [63][68]

Unpooling (obrázek **1.24** Ukázka max poolingu z obrázku 1.23) je vrstvou nadvzorkování, kterou lze ale použít pouze v případě, že byla v rámci sítě využita vrstva pooling. Pro správnou funkci je nutné si pamatovat pozice nalezených maxim během

poolingu, v případě, že ve vrstvách pro podvzorkování byla přítomna vrstva max poolingu. [63][64]



Obrázek 1.24 Ukázka max poolingu z obrázku 1.23 a unpoolingu, velikost okna a jeho krok je stejný jako na obrázku 1.23 (upraveno z [64])

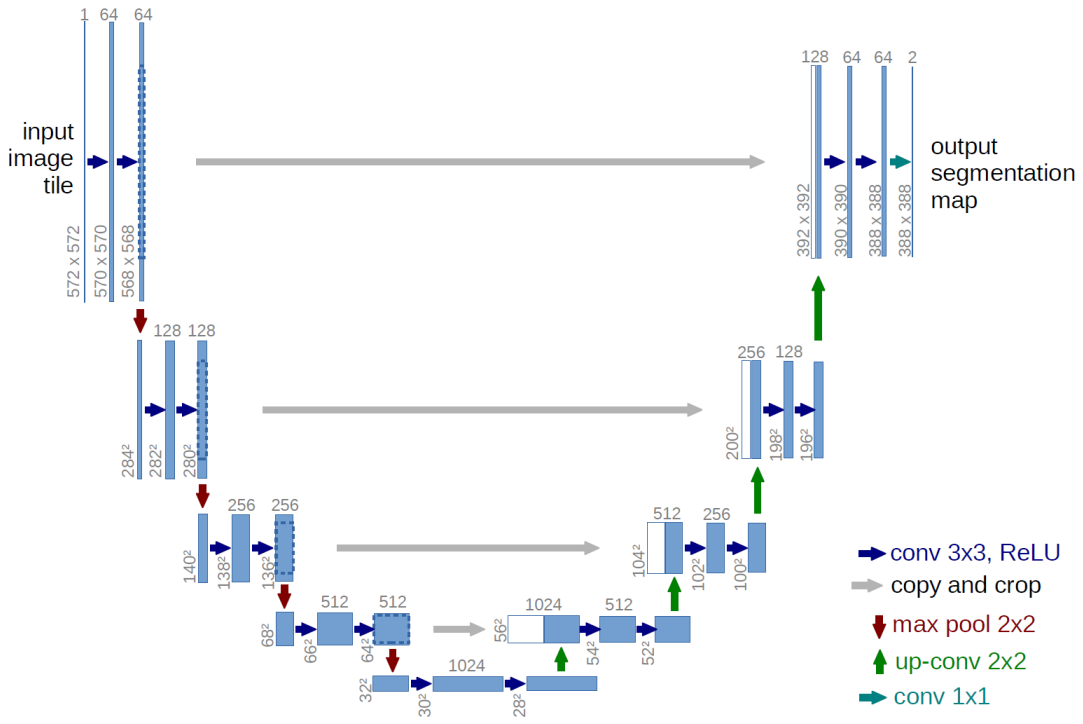
Transponovaná konvoluce, nejedná se o inverzi k operaci konvoluce. Slouží k nadvzorkování výstupu z předchozí vrstvy pomocí maticového násobení. Váhy masek se učí. [63][68]

1.6.8 Architektury sítě

Různým uspořádáním jednotlivých vrstev lze dosáhnout různých topologií, tedy jiných architektur. Mezi známé architektury patří například LeNet, AlexNet, U-net či SegNet. [69][70][71][72]

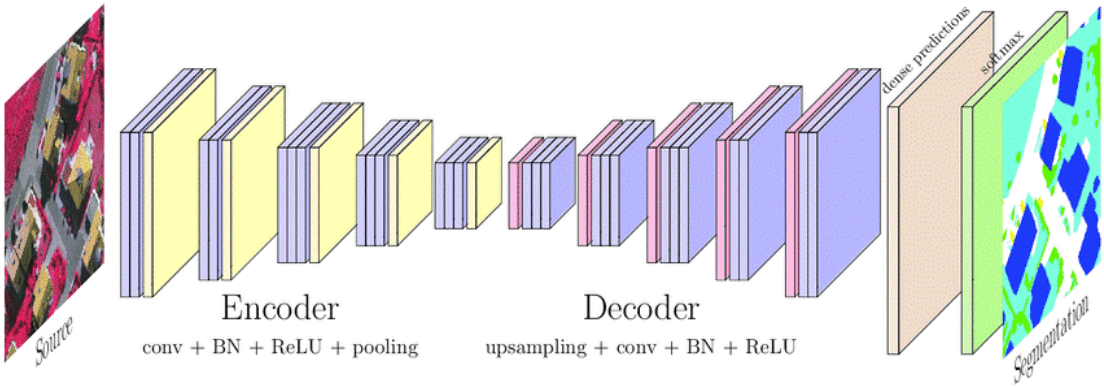
U-net byl vyvinut týmem vědců na ústavu počítačových věd ve Freiburské univerzitě roku 2015. Byla určena původně pro sémantickou segmentaci biomedicínských obrazů, ale uplatnila se i u nemedicínské problematiky. Hlavní výhodou této architektury je schopnost správného natrénovaní i s menším trénovací množinou dat (desítky obrazů). [71][73] Název U-net vyplývá z tvaru architektury, která je ilustrovaná na obrázku 1.25.

Síť je tvořena tzv. kontrahující cestou, tedy encoderem (extraktor příznaků a zároveň podvzorkování – levá strana sítě) a cestou expansivní, dekodérem, (část nadvzorkování – pravá strana sítě), přičemž tyto cesty jsou propojeny (skip connections) za využití konkatenace (šedé šipky). Encoder obsahuje 3x3 konvoluční vrstvy, kdy každá tato vrstva je následována aktivační funkcí ReLU (modré šipky) a 2x2 pooling vrstvami (červené šipky). V kontrahující části nejdříve nastává nadvzorkování (transponovaná konvoluce) a následuje 3x3 konvoluce s aktivační funkcí ReLU. [71]



Obrázek 1.25 Architektura U-net [71]

SegNet (obrázek **1.26**) je vhodná i pro instanční segmetnaci, topologie je podobná U-net. Hlavním rozdílem je absence přeskoků (skip connections) do hlubších vrstev. Dále je v architektuře dané, že síť je následována aktivační funkcí ReLU až po regularizační Batch Norm vrstvě. Posledními vrstvami jsou plně propojená vrstva, což je plně propojená umělá neuronová síť a aktivační funkce SoftMax, která řeší klasifikaci do více tříd. [72]



Obrázek 1.26 Architektura SegNet [72]

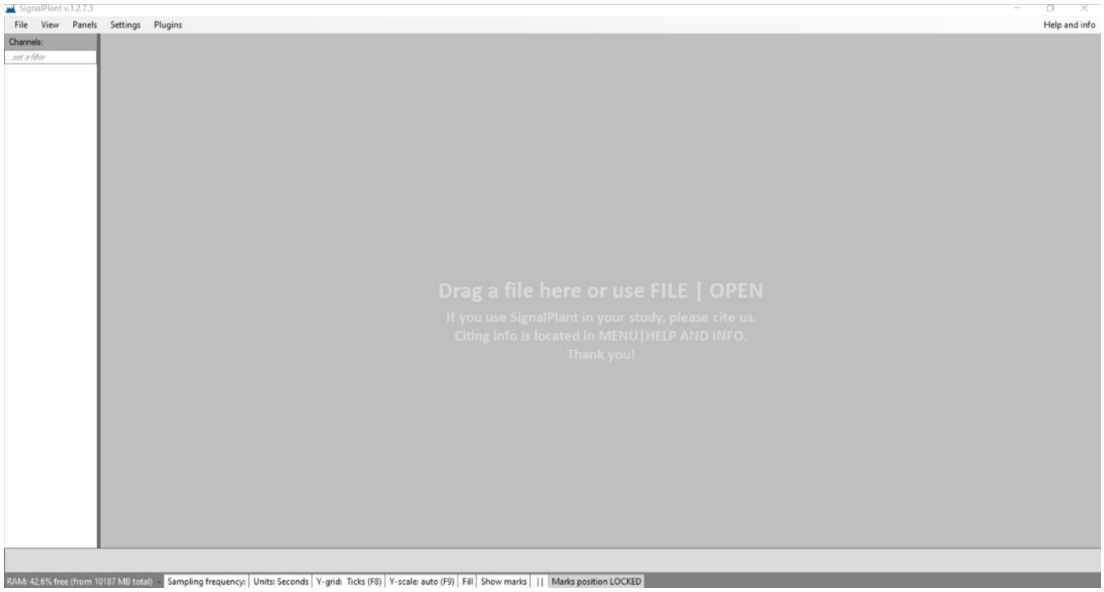
2. PRAKTICKÁ ČÁST

V rámci této práce byl pro realizaci modelu vybrán přístup pro segmentaci síňové aktivity, inspirovaný článkem [74], který se věnoval sémantické segmentaci EKG signálu nikoliv v 1D oblasti, nýbrž ve 2D za využití hluboké 2D konvoluční neuronové sítě. Toho bylo docíleno transformováním 1D signálu na 2D obraz (popsáno dále). Praktická část je tvořena několika částmi, a to ruční anotací, realizací modelu, které je dále pak tvořeno předzpracováním dat, tedy vytvořením 2D obrazů z 1D signálů a anotací, vytvořením modelu, jeho aplikováním a dodatečné úpravy výstupu. Předposlední částí je provedeno zhodnocení modelu, interpretace a diskuze výsledků, a nakonec je porovnáno řešení se zahraničními studiemi, věnující se podobnému tématu.

V této práci byly využity signály z Dětské nemocnice FN Brno, jejichž vzorkovací frekvence byly 2000 Hz a délka se pohybovala kolem 11000 vzorků (záznamy byly různě dlouhé). Data byla poskytnuta se souhlasem etické komise a pacienti podepsali informovaný souhlas s využitím dat pro vědecké účely.

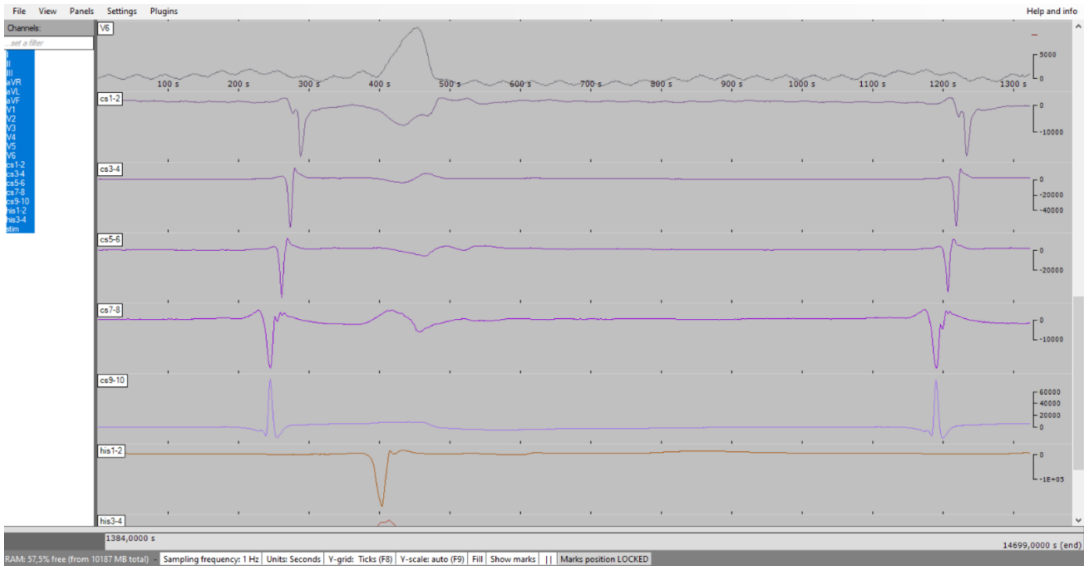
2.1 Anotace iEKG signálu

Pro ruční anotaci intrakardiálního EKG signálu byl využitý softwarový nástroj SignalPlant v.1.2.7.3, který je zdarma a byl vyvinutý oddělením medicínských signálů z Ústavu přístroje techniky Akademie věd České republiky, v.v.i.. Vzhled grafického uživatelského rozhraní je ilustrován na obrázku **2.1** níže.



Obrázek 2.1 Grafické uživatelské rozhraní SignalPlant

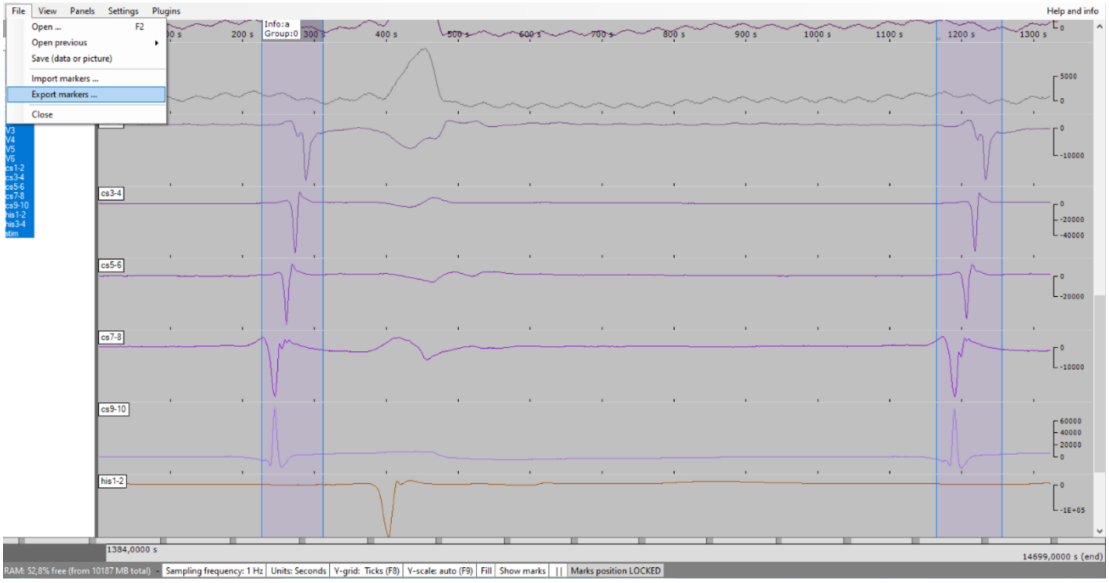
Intrakardiální záznamy byly ve formátu .csv. Záznam stačí pouze přesunout myší do programu a sám se otevřel. Na obrázku **2.2** je zobrazen jeden z nahraných záznamů, kde si lze povšimnout například záznamů z kanálů CS1-2 až CS9-10 (CS – snímání z koronárního sinu), kterým bude věnována v této práci pozornost.



Obrázek 2.2 Ukázka nahraného záznamu v prostředí SignalPlant

Ruční anotace se provede podržením klávesy shift a současně tažením levým tlačítkem myši, kterou vybereme požadovanou oblast. V případě této práce byla vybírána aktivita síní.

Po vybrání se zobrazí okno pro editaci výběru. V něm lze například upravit rozsah (uvedeno ve vzorcích), pojmenovat výběr (Additional information) a přiřadit tento výběr svodu. Jelikož byly anotace provedeny pro všechny kanály, nebylo potřeba k těmto výběrům přiřazovat konkrétní svody. Anotace se nakonec exportují ve file -> export markers (obrázek **2.3**).



Obrázek 2.3 Export anotací

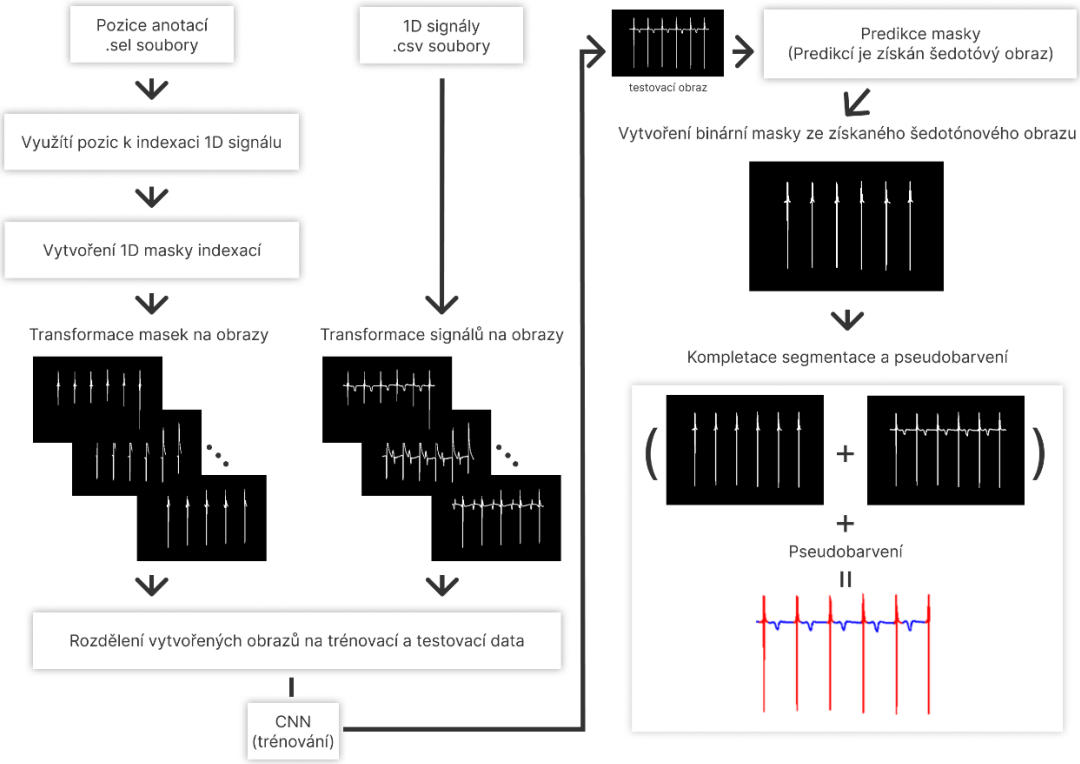
Exportem vznikne .sel soubor, který obsahuje pozice jednotlivých anotací v rámci záznamu. Tyto pozice jsou udávány ve formě „od vzorku ke vzorku“. Jedním z cílů dalšího kroku, tedy předzpracování, bude tyto údaje z tohoto souboru správně extrahovat, aby bylo možné pomocí těchto dat vytvořit binární masky.

2.2 Realizace modelu

Model byl realizován v programovacím jazyce Python ve vývojovém prostředí Spyder, na zařízení s 64bitovým operačním systémem Windows, RAM 12 GB a procesorem AMD Ryzen 5 3500U s integrovanou grafickou kartou Radeon Vega Mobile Gfx s taktovací frekvencí 2.10 GHz. Bylo použito několik knihoven, které bylo potřeba nainstalovat (použité knihovny jsou vypsány v příloze, v souboru README).

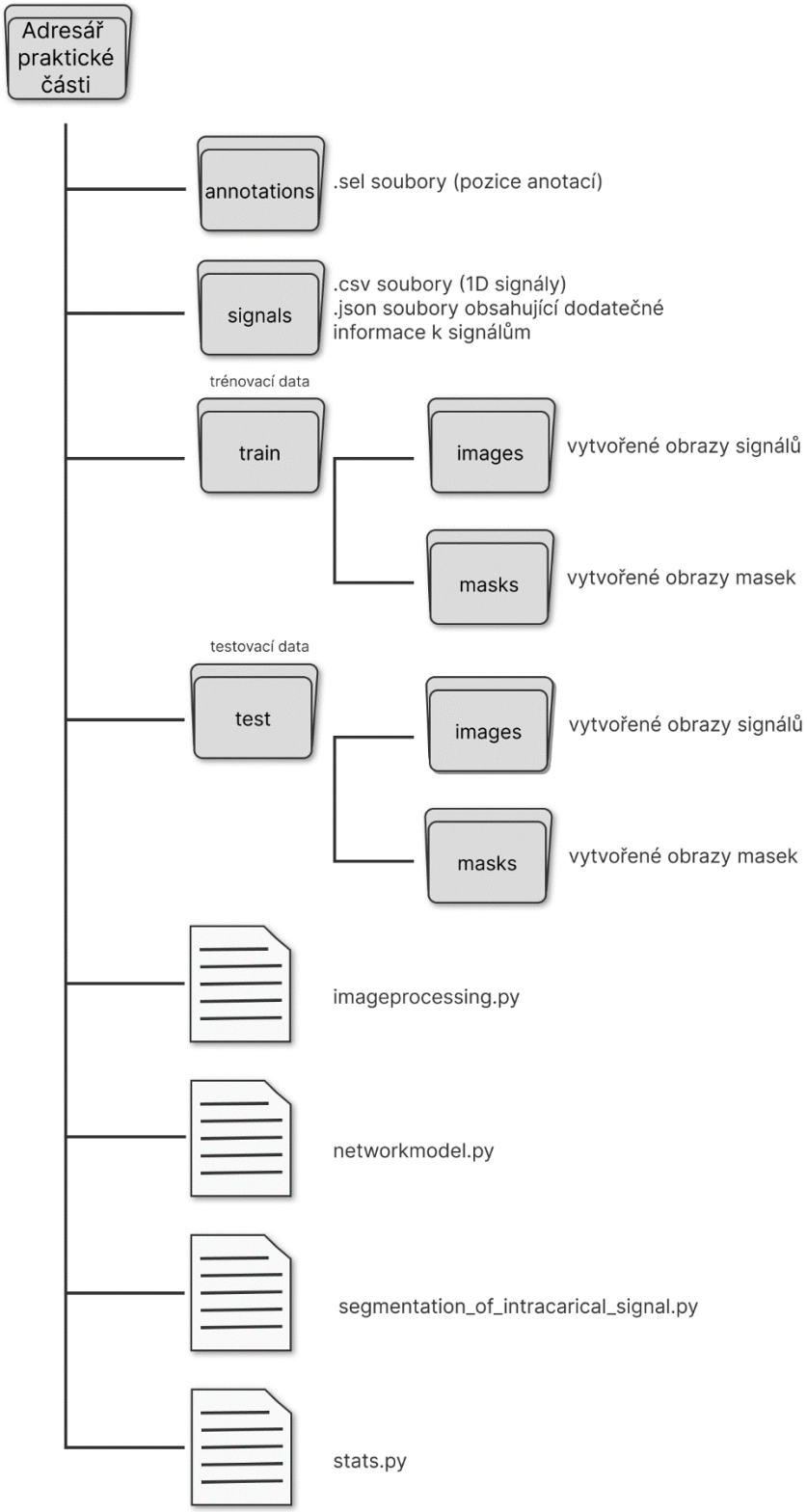
Jak již bylo zmíněno v úvodu praktické části, segmentace záznamů nebyla provedena v 1D oblasti, ale ve 2D za pomocí konvoluční neuronové sítě a číslicového zpracování obrazů.

Prvním krokem bylo nahrání a předzpracování dat do podoby, aby byla síť schopna s těmito daty pracovat jako s obrazy. Bylo tedy nutné 1D signály a anotace převést na obrazy. Následně se vytvořila konvoluční neuronová síť, kdy pro trénování a predikci byly využity vytvořené obrazy. Predikcí modelu byla získána šedotónová maska, která reprezentovala hledanou oblast. Nakonec byla provedena kompletace segmentace spojením upravené predikované masky a obrazu signálu a pseudobarvením. Proces realizace modelu je ilustrován na schématu níže (obrázek **2.4**). V následujícím textu budou jednotlivé kroky popsány podrobněji.



Obrázek 2.4 Schéma realizace řešení pro segmentaci intrakardiálního záznamu ve 2D

Pro realizaci praktické části (které je schématicky ilustrováno na obrázku **2.4**) bylo navrhnuto a realizováno řešení, jejíž struktura je na obrázku **2.5**, který se nachází na další straně. Adresář *annotations* obsahuje .sel soubory. Adresář *signals* obsahuje .csv soubory obsahující intrakardiální záznamy a soubory .json, které obsahují dodatečné informace jako například vzorkovací frekvenci, věk a pohlaví pacienta, délku záznamu, autor měření či diagnosu pacienta. Soubor *imageprocessing.py* je soubor, který obsahuje vytvořenou stejnojmennou třídu `ImageProcessing`, obsahující metody určené pro zpracování obrazu. Nachází se v ní například metoda pro pseudobarvení, normalizaci a dilataci obrazu či pro vytvoření obrazu z 1D záznamů. V dalším vytvořeném souboru, *networkmodel.py*, se také nachází stejnojmenná třída `NetworkModel`, v níž se nachází metody určené k vytvoření konvolučních neuronových sítí. V této práci byly zvoleny architektury U-net a SegNet, které byly pro potřeby této práce upraveny (více bude popsáno níže). Předposlední soubor *stats.py*, v níž je třída `Stats`, obsahuje metody pro statistiku, která slouží k zhodnocení modelu. Posledním Python soubor, *segmentation_of_intracardical_signal.py*, je hlavní soubor. Jedná se o jediný soubor, který se spouští. Obsahuje konfiguraci modelu, jako například na stavení cest, vymezení délky signálů, poměr rozdělení vytvořených obrazů do trénovací a testovací množiny dat, počet epoch a další. Také v ní probíhá nahrání dat, propojení metod ze všech tříd a kompletace segmentace.



Obrázek 2.5 Struktura řešení praktické části

Dále byly v adresáři řešení přítomny další soubory jako .h5 soubor(y), obsahující váhy a konfiguraci modelu, které si knihovna Keras vytváří, případně přeukládá, sama (zmíněno dále) a soubory *README_CZ.txt* a *README_ENG.txt*, které obsahují informace programu a instrukce o jeho správnému používání. Tyto soubory nebyly ve schématu ilustrovány, jelikož se nejedná o soubory řešení, ale jen dodatečné soubory, a tak na její realizaci neměly vliv.

2.2.1 Kontrola a předzpracování dat

Nejdříve bylo potřeba získat data z .csv a .sel souborů, ze kterých se pak mohly vytvořit obrazy. Proces nahrání probíhal tak, že se nejdříve použila metoda `listdir()`, kde vstupem byla cesta ke složce s daty, například adresář se signály. Tato metoda vracela datovou strukturu list, obsahující názvy souborů ve složce. Na tento list se použil cyklus for, ve kterém se provedlo předzpracování. Jako dočasnou hodnotou cyklu for byla proměnná `file`, která v dané iteraci obsahovala název aktuálního souboru. Soubory .csv a jejich příslušné .sel soubory měli stejné názvy. Dříve, než se nahrály soubory, provedla se kontrola, zda-li soubory existují. Mohl totiž nastat případ, že může v rámci ”páru” signálu (.csv) a anotace (.sel) mohl existovat pouze jeden z těchto souborů, což by mohlo zkomplikovat správnou funkčnost kódu. Nastal-li takový případ, kód tento pár ignoroval přeskočením do další iterace přes `continue`. Pokud .csv a příslušný .sel soubor existoval, přeskočení nenastalo a program pokračoval dále.

Pokud bylo vše v pořádku, po této kontrole následovalo nahraní dat. Soubor .csv, obsahující bio-signály, se importovaly pomocí funkce `read_csv()` z knihovny Pandas. Návratovou hodnotou byl *dataframe*, který obsahoval signály ze všech svodů. V rámci práce byla věnována pozornost CS svodům (CS1-2 až CS9-10). Svody byly ve sloupcích, vybraly se tedy pouze potřebné sloupce. Pro další práci bylo vhodné pracovat s datovou strukturou *array*, bylo tedy nutné převést *DataFrame* do *array*. Ten byl o rozměrech N x 5, kde N byl počet vzorků a číslo 5 byl počet kanálů. *Array* se následně transponoval, aby byl ve tvaru 5 x N. Následně se přidal pomocí metody `append()` do listu `data_array_plot`, který sloužil k uložení všech záznamů z .csv souborů.

Po nahrání .csv přišel na řadu soubor .sel, kde, jak již bylo zmíněno dříve, obsahoval poziční informace o anotacích. Řádky ze souboru se extrahovaly do listu pomocí funkce `readlines()`. List nicméně obsahoval i nepotřebné řádky. U všech .sel souborů začínaly poziční informace na 31. řádku. Po pozičních informacích již nebyl přítomen další obsah, tudíž tyto údaje bylo možné získat obyčejným indexováním od 30. řádku (indexuje se od 0).

Díky datům z .sel souboru bylo následně možné vytvořit masku pro záznam (pro jeho příslušné signály). Ta se vytvořila prostým indexováním, kde se ze .sel souboru nalezly počátky a konce anotací, které se použili na indexaci iEKG signálu ve všech 5ti svodech. Maska tedy měla stejné rozměry jako *array* vytvořený z .csv souboru, tedy 5 x N.

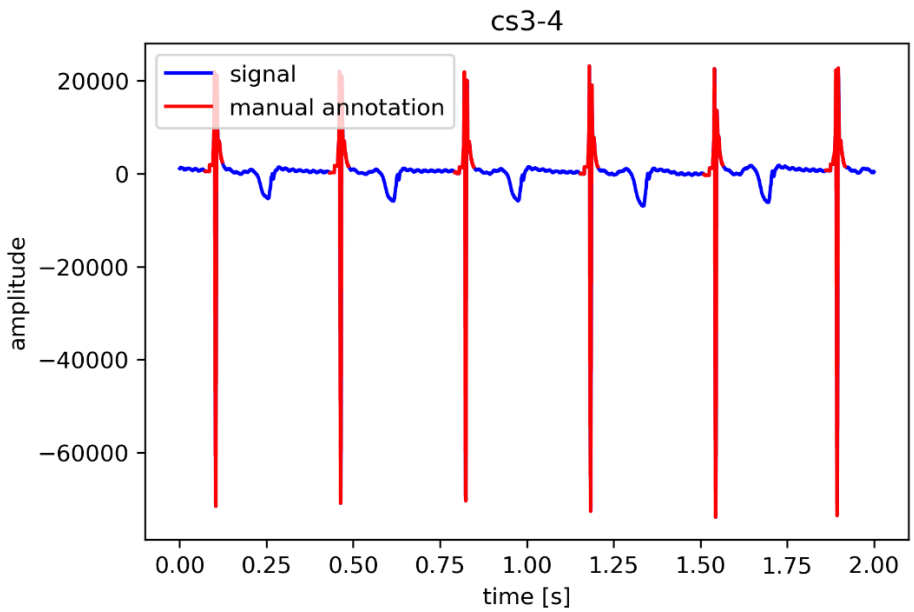
Nakonec, stejně jako array s iEKG signály, se v rámci iterace přidala do listu `mask_array_plot`.

Aby nebyly jednotlivé záznamy různě dlouhé, ustálila se délka signálů (a tím i korespondujících masek) na jednotnou délku 4000 vzorků (2s).

Jako prevence proti fragmentaci paměti[2] se na konci for cyklu zavolala funkce `collect()` z modulu Garbage Collector.

Po dokončení cyklu for byly listy `mask_array_plot`, obsahující masky (indexované úseky signálu dle pozic anotací) a `data_array_plot`, obsahující záznamy pacientů (každý záznam obsahoval signály z 5ti svodů) na další krok.

V rámci procesu popsaného výše bylo provedeno nahrání dat ze souboru .json, konkrétně vzorkovací frekvence. Ačkoli je tento proces v kódu ještě před nahráním .csv a .sel souborů, byl tento krok popsán až nyní, jelikož v rámci konfigurace je možné nastavit v základu vzorkovací frekvenci ručně. Extrahování ze vzorkovací frekvence bylo provedeno nahráním souboru přes funkci `open()`, parsováním dat přes `json.loads()` čímže došlo k převodu dat z JSON do asociativního pole. Indexace probíhá pomocí klíče, který oproti běžným polím (array) není číselný, ale jako textový řetězec. Tyto informace byly uloženy do proměnné `json_data`. Vzorkovací frekvence byla nakonec získána indexací z tohoto pole, kde klíči byly v tomto případě `'amp_settings'` a `'sampling_f'`. Náhled signálu s manuální anotací je obrázku **2.6**.



Obrázek 2.6 Náhled nahraného iEKG signálu s manuální anotací

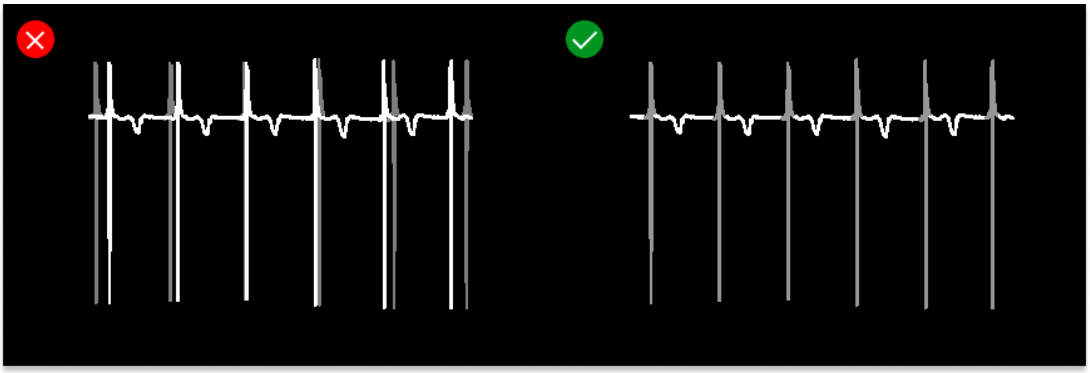
[2] Fragmentace paměti je situace, kdy je paměť neefektivně využívaná, a to ukládáním dat po částech v nesouvislém uspořádání [75]

2.2.2 Transformování 1D signálu a masky na 2D obrazy

Listy `mask_array_plot` a `data_array_plot` z předchozího kroku byly využity pro vytvoření 2D obrazů, které budou následně sloužit jako datové soubory pro konvoluční neuronovou síť. Proces probíhal opět skrze cyklus for, kdy byly procházeny oba dva listy. Jelikož měly stejnou délku, stačilo pro určení počtu iterací použit libovolně jeden z nich. V rámci iterace probíhalo vytváření obrazů signálů a příslušných masek.

Nejdříve se však automaticky určilo, kam se vytvořený obraz přiřadí, tedy, zdali do složky pro trénovací nebo testovací data. Jinými slovy, určila se cesta pro uložení vytvořeného obrazu. Vytvořil se pro tyto účely textový řetězec, který měl podobu `'M:N'`, kdy M a N jsou celá čísla dělitelná deseti. Jednalo se o poměr trénovacích a testovacích dat, kdy M reprezentoval procentuální zastoupení trénovacích tedy, kolik procent ze všech vytvořených obrazů se nahraje do složky pro trénovací data. Pro N platilo analogicky to samé pro testovací data. Textový řetězec se pomocí `split()` rozdělil na array (separačním znakem byla dvojtečka), který měl na první pozici M a na druhé N. Obě tyto hodnoty se převedly na datový typ integer (celá čísla). Tyto hodnoty se vložily do podmínek, které měly na starost právě samotné určení cesty.

Po přiřazení cesty probíhá již samostatný proces vytvoření obrazu a uložení. Jelikož zde barevná informace nebyla důležitá, vytvořené obrazy signálů a masek byly binární. Cílem tohoto kroku bylo tedy vytvoření binárních obrazů masek a signálů, kde obrazy signálu a jeho příslušné masky na sebe musí pasovat (obrázek **2.7** - vpravo). Přesněji řečeno, aby maska, respektive jeho bílé pixely, odpovídaly přesně příslušnému úseku bílých pixelů obrazu signálu.



Obrázek 2.7 Ilustrativní náhled správného (vpravo) a chybného (vlevo), vytvoření obrazů signálu (bílá) a příslušné masky (šedá)

Byl vymyšlený následující postup: vykreslil se signál s manuální anotací, tak jako na obrázku 2.6, kdy pro oblast masky se přesně specifikovala červená barva (*#ff0000*) a pro oblast signál modrá barva (*#0000ff*). Následně bylo potřeba odstranit hodnoty u osy x a y a rám. Hodnoty u jednotlivých os se schovaly přes `plt.xticks([])` a `plt.yticks([])`. Rám se nastavil vypnutím viditelnosti přes funkci `set_visible()`, kde se do závorek dala hodnota `False`. S tímto vykreslením bylo

v dalším kroku potřeba pracovat už jako s obrazem. Vytvořil se tedy dočasný barevný RGB obraz přes funkci `savefig()`. Tento obraz byl důležitý z hlediska jeho barevné informace, jelikož díky této informaci se pak vědělo, kde v obraze se nachází masky a kde signál. Aby se zachovalo co nejvíce správné barevné informace, přidal se do funkce `savefig()` kromě cesty a názvu uloženého obrazu i parametr `dpi`[3] na hodnotu 600, aby byl obraz co nejkvalitnější. Kvalitním obrazem tak bylo zajištěno, že (téměř všechny) pixely měly u vytvořeného RGB obrazu správné stanovené hodnoty červené a modré. Nižším rozlišením by totiž některé pixely, u kterých byla požadovaná barva v hexadecimálním kódování *#ff0000*, tedy červená (v RGB [255, 0, 0]), mohly mít také červenou barvu, ale v jiném odstínu, například *#ed1b0c*, v RGB [237, 27, 12]. Tím by se proces vytvoření binárních obrazů signálů a příslušných masek mohl ztížit a bylo by potřeba myslet na širší rozsah odstínů jednotlivých barev. Vysokým DPI tak byl tento problém vyřešen. Vytvořený obraz se tak mohl nahrát pomocí funkce `imread()` z OpenCv a bylo tak možné se signálem pracovat už jako s obrazem. Z tohoto obrazu byl následně vytvořen binární obraz masky a příslušného signálu.

Jelikož signál zaujímal i oblast, kde se nacházela maska, bylo jeho vytvoření snadné. Stačilo pro vytvoření obrazu signálu převést nahraný RGB obraz na šedotónový přes `cvtColor()`, kde nejasnějších hodnot (bílá barva) disponovaly pixely pozadí. Provedl se negativ, čímž nejjasnějšími pixely byly pixely obrazu signálu a provedlo se prosté prahování. Vznikl pak binární obraz, kde bílé pixely reprezentovaly signál, zbytek byl černý.

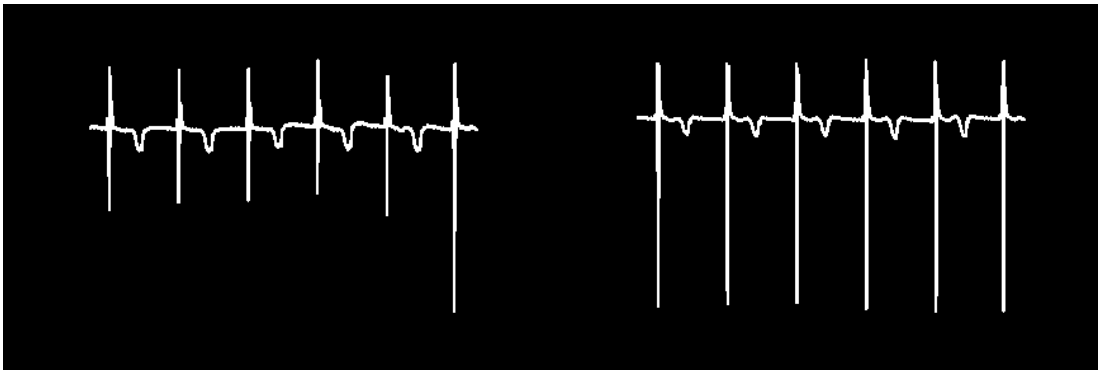
Vytvoření obrazu masky bylo mírně komplikovanější. U nahraného RGB obrazu se nalezly pouze červené pixely tedy s hodnotou v RGB [255, 0, 0] pomocí `inRange()` (OpenCv), čímže se získala maska (binární obraz). Pozice pixelů bílé barvy ale neseděly s bílými pixely obrazu signálů. Někde přesahovaly a někde naopak chyběly (horší případ). Bylo tedy nutné tyto struktury zvětšit pomocí morfologické operace dilatace přes funkci `dilate()` (OpenCv), kde se jako strukturní element použil čtverec o rozměrech 5x5. Ten se vytvořil přes funkci `ones()` (NumPy), čímž se vytvořila jedničková matice. Křivka masky byla širší než křivka signálu, bylo potřeba upravit křivku tak, aby byla stejná jako u obrazu signálu. To se vyřešilo průnikem obrazů. Dříve se ale obrazy masky a signálu se přeškolovaly přes `resize` (OpenCv), jelikož byly příliš velké a převedly se na binární obrazy prahováním, kde bílé pixely měly tentokrát hodnotu 1. Prahování se provedlo jednak kvůli interpolaci, která vznikla přeškálováním a také kvůli již zmíněnému průniku. Jelikož pixely disponovaly pouze hodnotami 0/1 a dilatovaný obraz masky přesahoval bílými pixely oblasti signálu, stačilo pouze pro provedení průniku provést vynásobení obrazu masky a obrazu signálu. Tím se u obrazu masky pixely s hodnotou 1 zachovaly pouze v místech, kde byly pixely s hodnotou 1 zároveň hodnoty

[3] DPI, dots per inch, udává počet pixelů na palec [76]

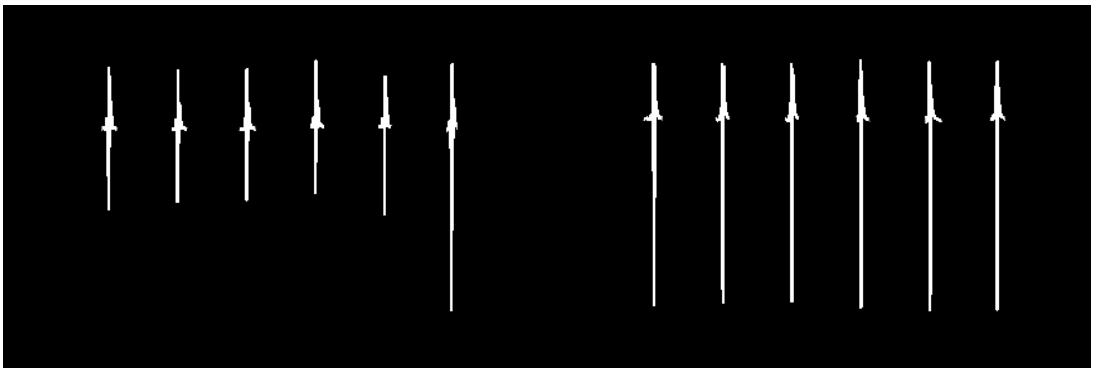
i u obrazu signálu. Obraz masky tak dobře lícoval k příslušnému obrazu signálu. Celý proces vytvoření je formou schématu ilustrován na obrázku **2.10**.

Před uložením bylo nutné změnit hodnoty pixelů z hodnoty 1 na hodnotu 255, jelikož obraz byl datového typu unit8 (0 až 255) a hodnota 1 by místo bílé barvy byla velmi tmavou (téměř černou) hodnotou šedi.

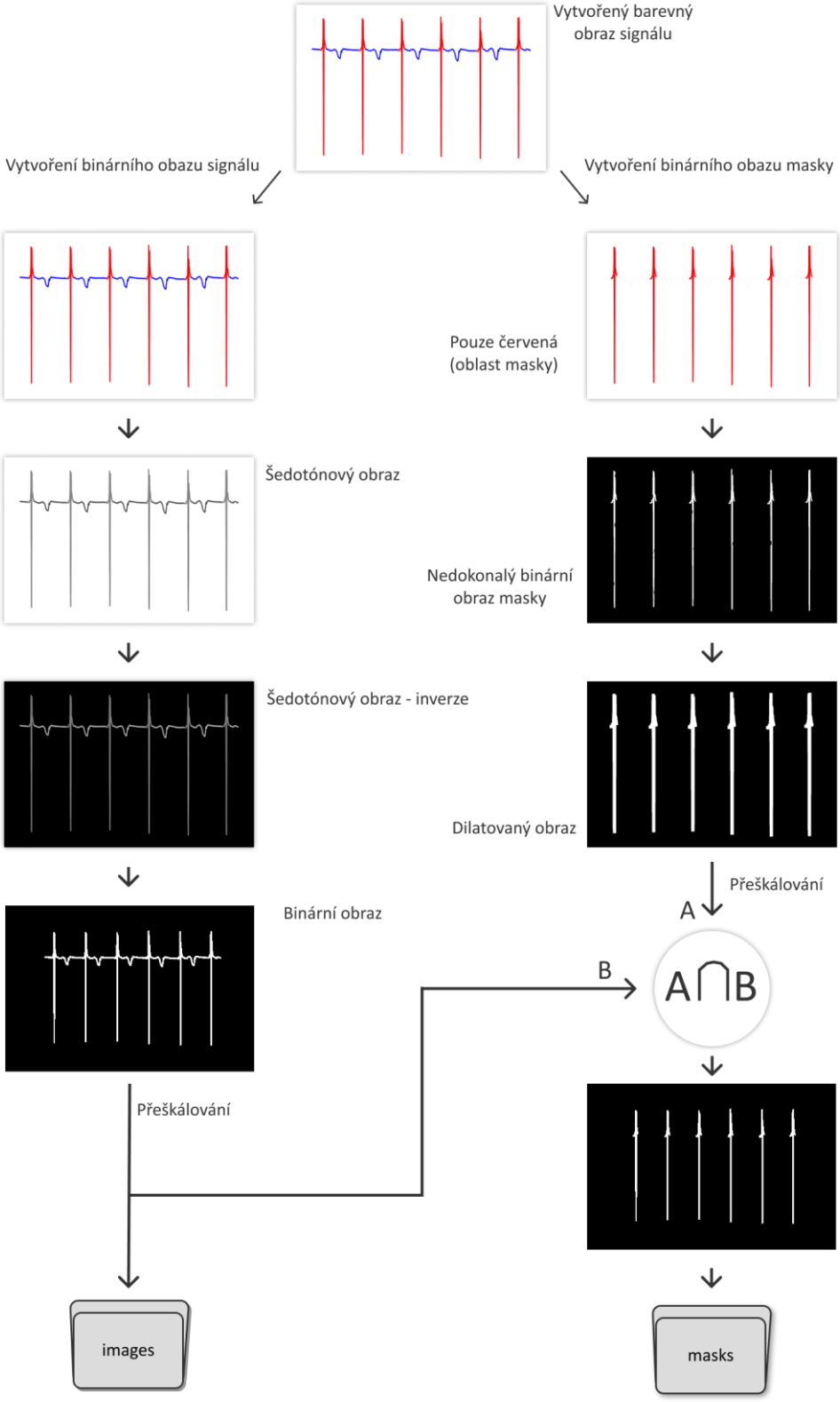
Nakonec se obrazy signálu a příslušené masky uložily pomocí metody `imwrite()` taktéž z knihovny OpenCv. Použil se stejný název jako u dočesného RGB obrázku, čímž se obraz nahradil požadovaným, binárním obrazem. Vytvořené obrazy masek sloužily jako label pro CNN. Na obrázku **2.8** a **2.9** jsou ukázky převedených signálů a příslušných masek na obrazy. Dále následuje vytvoření konvoluční neuronové sítě. V práci byly zvoleny U-net a SegNet.



Obrázek 2.8 Ukázka převedených signálů na binární obrazy



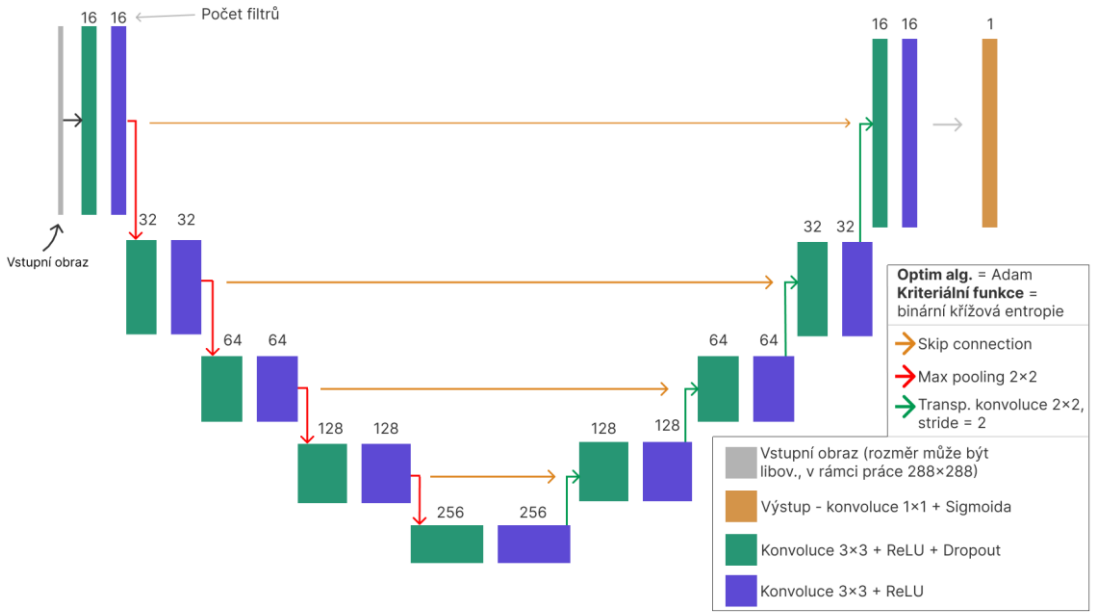
Obrázek 2.9 Ukázka převedených anotací na binární masky



Obrázek 2.10 Schéma procesu vytvoření (popsaný výše) binárního obrazu signálu (vlevo) a binárního obrazu masky (vpravo)

2.2.1 Realizace U-net

U-net, která byla v této práci použita, byla upravena. Pro vytvoření konvoluční neuronové sítě bylo využito knihovny TensorFlow [77], zejména pak jeho součásti Keras. [78] Kvůli větší přehlednosti byla vytvořena třída `NetworkModel` obsahující metody, které sloužily k vytvoření sítě, dle svého názvu, s příslušnou architekturou. Byla například vytvořena metoda pro architekturu U-net, jejíž topologie je na obrázku **2.11**.



Obrázek 2.11 Navrhnutá alternativa U-net

Do sítě vstupuje obraz, v řešení nebyla stanovena fixní hodnota (lze ji nastavit v konfiguraci v *segmentation_of_intracardical_signal.py*), pro práci bylo ale zvoleno 288x288. Levou stranu pak tvoří encoder, kde každý blok (patro) je tvořen dvěma konvolučními vrstvami o velikosti masek 3x3 s aktivačními funkce ReLU (na obraze **2.11** to jsou zelené a modré obdélníky). V Keras Tensorflow bylo možné vytvořit konvoluční vrstvu pomocí `Conv2D()`, kde bylo nutné definovat argumenty jako například počet filtrů, velikost masky, padding, nastavení vah, stride nebo možnost nastavit a specifikovat aktivační funkci. Počet filtrů s každým patrem narůstal od 16, 32, 64, 128 a 256 (decoder pak pokračuje od spodu, tedy od 256 filtrů). Velikost masky byla 3x3. Padding byl nastavený na `'same'`, výstup byl tak stejně velký jako vstup. Aktivační funkce byla zvolena již zmíněná ReLU. Byla taky možnost nastavit váhy kernelu, tedy mapy příznaků přes argument `kernel_initilazer`, která stanovuje, jakou statistickou distribuci použít pro vytvoření vah. Dle článku [79], který se věnuje volbou inicializace vah, bylo nakonec u argumentu `kernel_initilazer` nastaveno `he_normal`, jelikož použitou aktivační funkcí bylo ReLU. Stride (krok filtru) se nechal na výchozím kroku 1. Vstup do této vrstvy (nejen do konvoluční vrstvy, ale do všech) se následně zapsal do závorek `Conv2D(argumenty)(vstup)`.

Mezi těmito konvolučními vrstvami se také nacházela regularizační vrstva dropout (na obrázku **2.11** součást zeleného obdélníku), která se vytvořila přes `Dropout()`, kde se nastavil tvz. dropout rate argument, tedy pravděpodobnost vypnutí. Bylo možné zvolit hodnotu od 0 do 1. Až na dno, kde byl dropout rate 0.3 byl u všech dropout vrstev nastavena hodnota 0.1.

Blok encoderu byl zakončen vrstvou max pooling (a obrázku **2.11** červená šipka), která se vytvořila přes `MaxPooling2D()`. Výstup z této vrstvy vstupoval do konvoluční vrstvy dalšího encoder bloku.

Bloky decoderu byly podobné, akorát mezi „patry“ se nenacházela pooling vrstva, ale vrstvy nadvzorkování přes transponovanou konvoluci (zelené šipky). Ta se vytvořila přes `Conv2DTranspose()`, kde se nastavily argumenty jako počet filtrů (stejný jako u konvolučních vrstev téhož patra), stride, který byl nastaven na hodnotu 2 a padding, který byl nastaven na `'same'`. Mezi bloky encoderu a decoderu (tedy téhoř patra) byl skip connection. Propojení bylo provodeno přes konkatenaci, kterou bylo možné vytvořit přes `concatenate()`, do nějž vstupoval list vstupu, což byly výstup bloku encoderu a výstup transponované konvoluce, která je součástí bloku příslušného decoderu.

Výstupní vrstvou pak byla konvoluční vrstva pouze s jedním filtrem a o velikosti masky 1x1.

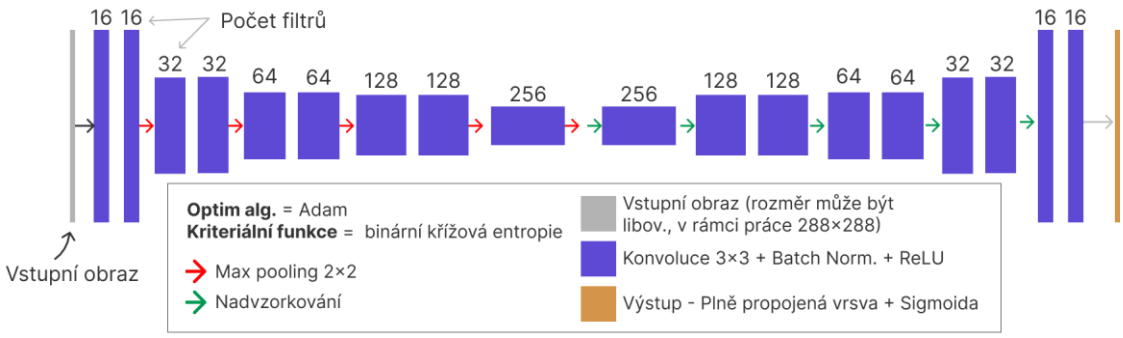
Nakonec se definoval optimalizační algoritmus a ztrátová funkce. Optimalizačním algoritmem byl zvolen Adaptive moment estimaition (Adam) a ztrátovou funkcí, jelikož se v této práci řeší binární klasifikace, byla zvolena binární křížová entropie a nastavení hodnotící metriky během trénování. To se definovalo pomocí metody `compile()`. Vytvořený model si lze textově vizualizovat na příkazové řádce pomocí metody `summary()`, náhled je na obrázku **2.12**.

```
concatenate_1 (Concatenate)     (None, 72, 72, 128)  0           conv2d_transpose_1[0][0]
                                                                 conv2d_5[0][0]
conv2d_12 (Conv2D)              (None, 72, 72, 64)   73792       concatenate_1[0][0]
dropout_6 (Dropout)             (None, 72, 72, 64)   0           conv2d_12[0][0]
conv2d_13 (Conv2D)              (None, 72, 72, 64)   36928       dropout_6[0][0]
conv2d_transpose_2 (Conv2DTrans (None, 144, 144, 32) 8224        conv2d_13[0][0]
concatenate_2 (Concatenate)     (None, 144, 144, 64) 0           conv2d_transpose_2[0][0]
                                                                 conv2d_3[0][0]
conv2d_14 (Conv2D)              (None, 144, 144, 32) 18464       concatenate_2[0][0]
dropout_7 (Dropout)             (None, 144, 144, 32) 0           conv2d_14[0][0]
conv2d_15 (Conv2D)              (None, 144, 144, 32) 9248        dropout_7[0][0]
conv2d_transpose_3 (Conv2DTrans (None, 288, 288, 16) 2064        conv2d_15[0][0]
concatenate_3 (Concatenate)     (None, 288, 288, 32) 0           conv2d_transpose_3[0][0]
                                                                 conv2d_1[0][0]
conv2d_16 (Conv2D)              (None, 288, 288, 16) 4624        concatenate_3[0][0]
dropout_8 (Dropout)             (None, 288, 288, 16) 0           conv2d_16[0][0]
conv2d_17 (Conv2D)              (None, 288, 288, 16) 2320        dropout_8[0][0]
conv2d_18 (Conv2D)              (None, 288, 288, 1)  17          conv2d_17[0][0]
==================================================================================================
Total params: 1,940,817
Trainable params: 1,940,817
Non-trainable params: 0
```

Obrázek 2.12 Souhrn vytvořeného U-net

2.2.2 Realizace SegNet

Stejně jako u U-net byl SegNet také nebyla využita přesná struktura a realizována rovněž za využití knihovny Keras TensorFlow. Navrhnutý SegNet je ilustrován na obrázku **2.13**.



Obrázek 2.13 Navrhnutá alternativa SegNet

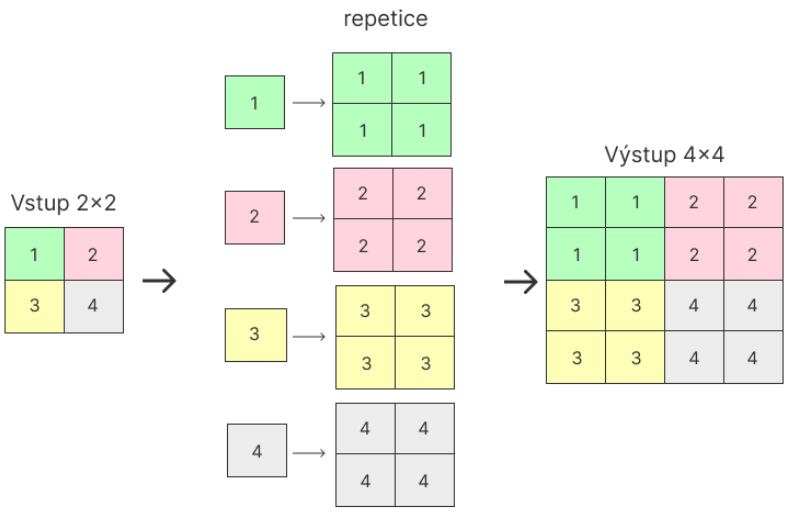
Disponuje podobnými vlastnostmi jako například volby stejného počtu filtrů (16, 32, 64, 128 a 256), velikosti kernelu u konvoluční a max pooling vrstvy (3x3 a 2x2), stejný padding u konvoluční vrstvy (`'same'`), volby aktivační funkce (ReLU), inicializátorem

vah (`'he_normal'`), optimalizačním algoritmem a ztrátové funkce. Sumarizace modelu na příkazové řádce je na obrázku **2.14**.

```
up_sampling2d_3 (UpSampling2 (None, 144, 144, 64)      0
conv2d_14 (Conv2D)           (None, 144, 144, 32)      18464
batch_normalization_12 (Batc (None, 144, 144, 32)      128
activation_12 (Activation)   (None, 144, 144, 32)      0
conv2d_15 (Conv2D)           (None, 144, 144, 32)      9248
batch_normalization_13 (Batc (None, 144, 144, 32)      128
activation_13 (Activation)   (None, 144, 144, 32)      0
up_sampling2d_4 (UpSampling2 (None, 288, 288, 32)      0
conv2d_16 (Conv2D)           (None, 288, 288, 16)      4624
batch_normalization_14 (Batc (None, 288, 288, 16)      64
activation_14 (Activation)   (None, 288, 288, 16)      0
conv2d_17 (Conv2D)           (None, 288, 288, 16)      2320
batch_normalization_15 (Batc (None, 288, 288, 16)      64
activation_15 (Activation)   (None, 288, 288, 16)      0
dense (Dense)                (None, 288, 288, 1)       17
=================================================================
Total params: 1,770,337
Trainable params: 1,768,417
Non-trainable params: 1,920
```

Obrázek 2.14 Souhrn vytvořeného SegNet

Oproti U-net ale nedisponuje skip connections. Další odlišností je vrstva regularizace, kdy místo Droupout vrsvy využita Batch normalization vrstvy, která se navíc nachází mezi konvoluční vrstvou a aktivační vrstvou (ReLU). Batch norm. vrstva se vytvořila pomocí `BatchNormalization()(vstup)`. Také se lišilo nadvzorkování. U architektury U-net bylo využito transponované konvoluce, zde bylo nadvzorkování provedeno pomocí `UpSampling2D()` s velikostí (nadvzorkovacím fakturem) 2x2, který se choval tak, že provádí repetice hodnot po řádcích a sloupcích. Přesněji řečeno, když byla výstupní matice 2x2 a bylo využito `UpSampling2D()` s faktorem také 2x2, byla výsledná matice 4x4, přičemž se pro doplnění chyběních hodnot použijí kopie hodnot ze vstupní matice. Zmíněný příklad je znázorněn na obrázku **2.15**. [80]



Obrázek 2.15 Princip UpSampling2D (upraveno z [80])

2.2.3 Segmentace pomocí konvoluční neuronové sítě

Vytvořením modelu následoval již proces segmentace. Nejdříve bylo provedeno nahrání trénovacího souboru dat, tedy, nahrání vytvořených obrazů signálů a masek z předchozího kroku. Obrazy trénovacích signálů se po nahrání vložily do listu `train_images` a masky do `train_masks`. Aby byla zajištěna jednotná velikost a poměr stran, byl zvolen fixní rozměr s čtvercovým poměrem stran. Po nahrání, ale před vložením nahraného obrazu do listu `train_images` (nebo `train_mask`), se provedlo oříznutí obrazu čtvercovým oknem, jehož středem byl střed obrazu a normalizace. Důvodem oříznutí oproti přeškálování byla prevence proti nežádoucím změnám pozičních souřadnic (že obraz bude "stlačen" z jedné ze stran). Normalizaci v tomto případě stačilo provést prostým prahováním, kde všechny nenulové pixely byly rovny hodnotě 1 jinak 0. Po nahrání trénovacích dat bylo potřeba převést listy na numpy array. Z nahraných dat se provedlo rozdělení ještě na trénovací a validační soubory dat pomocí `train_test_split()` z knihovny scikit-lear. Vstupem byly proměnné (numpy array) `train_images` a `train_mask`. Tato metoda následně ze těchto vstupu rozdělila datasey na trénovací (`x_train` - signály, `y_train` - masky) a validační (`x_val` - signály, `y_val` - masky).

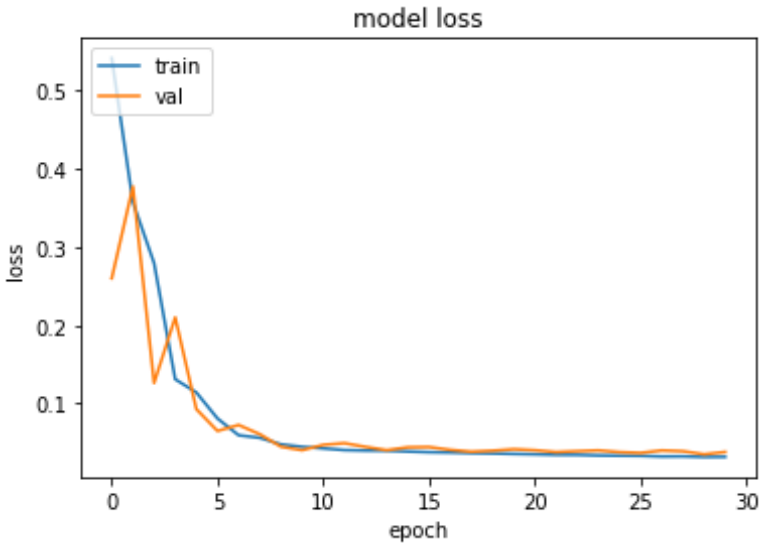
Dříve než se provedlo trénovaní, nastavilo se pomocí `EarlyStopping()`, kdy má model přerušit trénování (jak u U-net, tak i SegNet). To se provedlo vložením argumentů s hodnotami, které provedou požadovanou akci. Nastavilo se tedy, že model bude sledovat průběh validační chyby, nastavením argumentu `monitor='val_loss'`, a pokud nedošlo ke změně (ke zlepšení) ani po *N* epochách (`patience = N`, kde *N* je integer), model učení ukončil a obnovily se jen ty váhy, které z hlediska metrik poskytovaly nejlepší kvalitu (přes `restore_best_weights=True`).

Dále následovalo již samostatné trénování. Zde bylo nutné definovat počet epoch, dále `batch_size`, který udával počet trénovacích vzorků během iterace a takzvaný verbose, který sloužil pro zobrazení průběhu trénovaní v konzoli (verbose lze přeložit jako výřečnost). Verbose byl zde integer, který mohl nabývat hodnot 0, 1 nebo 2, kdy hodnota 0 nezobrazuje nic, hodnota 1 zobrazuje ukazatel průběhu (progress bar) a počet epoch (obrázek **2.16**), hodnota 2 zobrazuje pouze počet epoch. Byla tedy zvolena hodnota 1. Terénování pak bylo provedeno pomocí metody `fit()`, do jehož argumenty byla trénovací a validační množina dat, velikost batch, počet epoch a již zmíněný verbose. Bylo možné si volání této metody uložit do proměnné historii vývoje chyby a přesnosti.

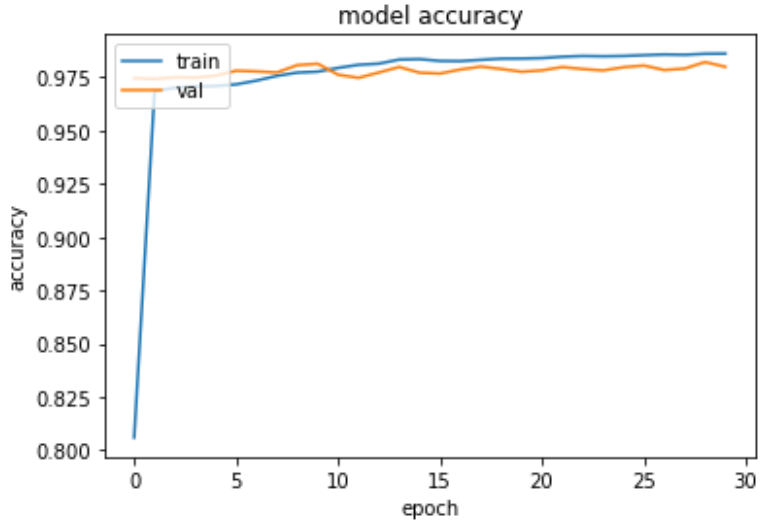
```
Epoch 14/30
1/1 [==============================] - 2s 2s/step - loss: 0.0710 - accuracy: 0.9735 - val_loss: 0.0670 - val_accuracy: 0.9798
Epoch 15/30
1/1 [==============================] - 2s 2s/step - loss: 0.0576 - accuracy: 0.9740 - val_loss: 0.0462 - val_accuracy: 0.9800
Epoch 16/30
1/1 [==============================] - 2s 2s/step - loss: 0.0797 - accuracy: 0.9721 - val_loss: 0.0698 - val_accuracy: 0.9821
Epoch 17/30
1/1 [==============================] - 2s 2s/step - loss: 0.0588 - accuracy: 0.9766 - val_loss: 0.0798 - val_accuracy: 0.9797
Epoch 18/30
1/1 [==============================] - 2s 2s/step - loss: 0.0665 - accuracy: 0.9761 - val_loss: 0.0758 - val_accuracy: 0.9789
Epoch 19/30
1/1 [==============================] - 2s 2s/step - loss: 0.0639 - accuracy: 0.9764 - val_loss: 0.0635 - val_accuracy: 0.9805
Epoch 20/30
1/1 [==============================] - 2s 2s/step - loss: 0.0548 - accuracy: 0.9775 - val_loss: 0.0490 - val_accuracy: 0.9825
Epoch 21/30
1/1 [==============================] - 2s 2s/step - loss: 0.0445 - accuracy: 0.9791 - val_loss: 0.0375 - val_accuracy: 0.9828
Epoch 22/30
1/1 [==============================] - 2s 2s/step - loss: 0.0433 - accuracy: 0.9777 - val_loss: 0.0348 - val_accuracy: 0.9828
Epoch 23/30
1/1 [==============================] - 2s 2s/step - loss: 0.0531 - accuracy: 0.9758 - val_loss: 0.0358 - val_accuracy: 0.9834
Epoch 24/30
1/1 [==============================] - 2s 2s/step - loss: 0.0400 - accuracy: 0.9793 - val_loss: 0.0394 - val_accuracy: 0.9833
Epoch 25/30
1/1 [==============================] - 2s 2s/step - loss: 0.0385 - accuracy: 0.9811 - val_loss: 0.0422 - val_accuracy: 0.9827
Epoch 26/30
1/1 [==============================] - 2s 2s/step - loss: 0.0398 - accuracy: 0.9811 - val_loss: 0.0432 - val_accuracy: 0.9821
Epoch 27/30
1/1 [==============================] - 2s 2s/step - loss: 0.0403 - accuracy: 0.9811 - val_loss: 0.0425 - val_accuracy: 0.9822
Epoch 28/30
1/1 [==============================] - 2s 2s/step - loss: 0.0395 - accuracy: 0.9816 - val_loss: 0.0402 - val_accuracy: 0.9829
Epoch 29/30
1/1 [==============================] - 2s 2s/step - loss: 0.0379 - accuracy: 0.9825 - val_loss: 0.0369 - val_accuracy: 0.9841
Epoch 30/30
1/1 [==============================] - 2s 2s/step - loss: 0.0363 - accuracy: 0.9833 - val_loss: 0.0336 - val_accuracy: 0.9855
```

Obrázek 2.16 Průběh učení, který je vyspaný na příkazové řádce, bývá vypsán, jak počet zbývajících epoch, tak i například vývoj trénovací a validační chyby

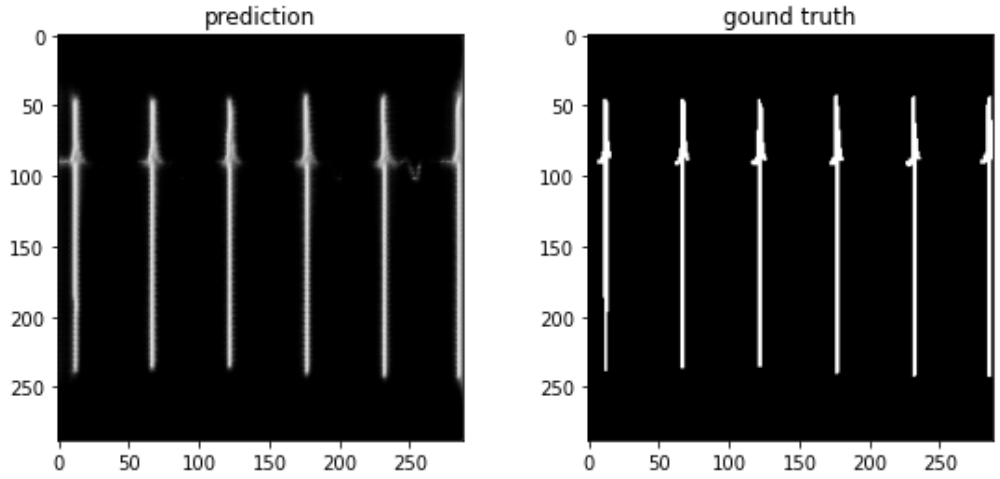
Pokud byl proces učení úspěšně dokončen, bylo možné uložit parametry modelu pomocí `model.save('libovolny_nazev_souboru.h5')`. Po natrénování bylo možné provést predikci, kdy se nahrál testovací obrázek (a ořízl na potřebný rozměr) a provedla se predikce pomocí `predict()`. Bylo možné si rovněž zobrazit vývoj chyby a přesnosti, které jsou k nahlédnutí na obrázku **2.17** a **2.18**. Výstup z predikce a jeho porovnaní s ground truth je zobrazen na obrázku **2.19**.



Obrázek 2.17 Vývoj trénovací (modrá křivka) a validační (oranžová křivka) chyby v epochách



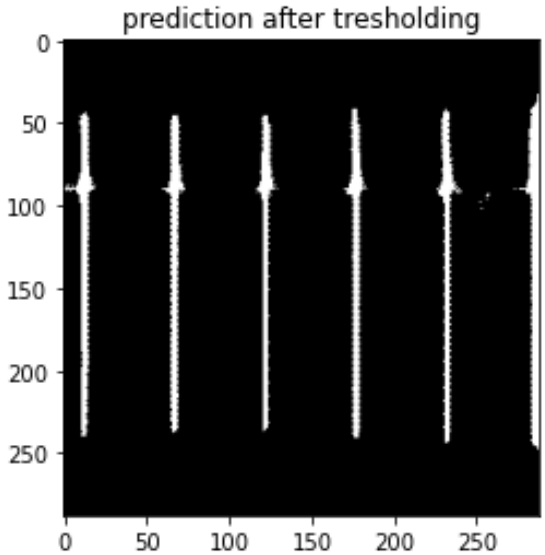
Obrázek 2.18 Vývoj trénovací (modrá křivka) a validační (oranžová křivka) přesnosti v epochách



Obrázek 2.19 Vlevo – predikce masky, vpravo – požadovaná maska

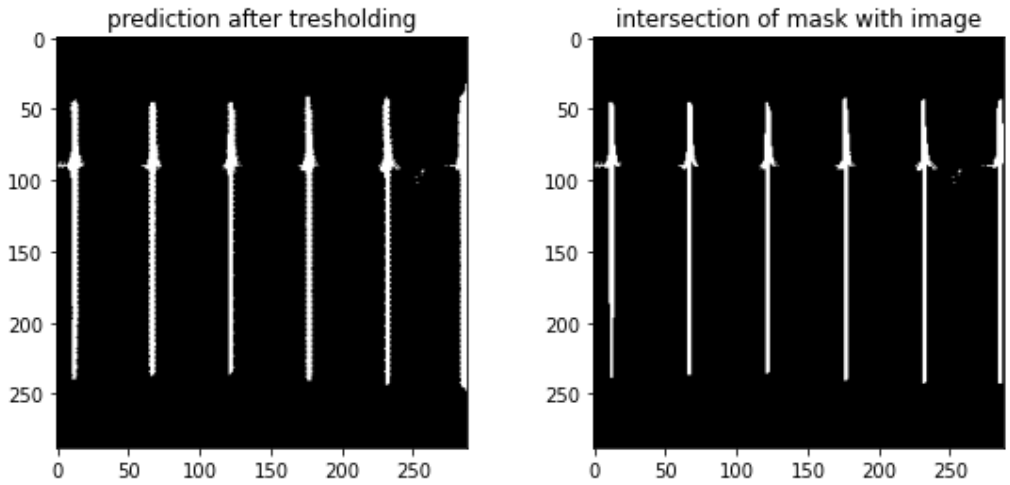
Predikovanou maskou následovala kompletace segmentace, která sloužila spíše pro lepší vizuální vjem. Výstupem z predikce byl šedotónový obraz. Bylo nutné z něj udělat obraz binární. Bylo využito prostého prahování (obrázek **2.20**), kde bylo využito automatického nalezení prahu pomocí metody Otsu, aby byl proces segmentace plně automatizován a nebylo tak potřeba nastavovat práh ručně.

Pro určení prahu se nevyužil celý obraz, nýbrž jen její oříznutá část. Mohla totiž nastat situace, že pixely s nejvyššími hodnotami šedi mohly být na okrajích. Automatickým hledáním prahu tak mohly být nesprávně zohledněny tyto intenzity a následným prahovaním by tak byly nenulové pouze okraje. Zbylé pixely by měly hodnotu 0. Nalezení prahu v oříznuté části obrazu toto riziko snížilo.



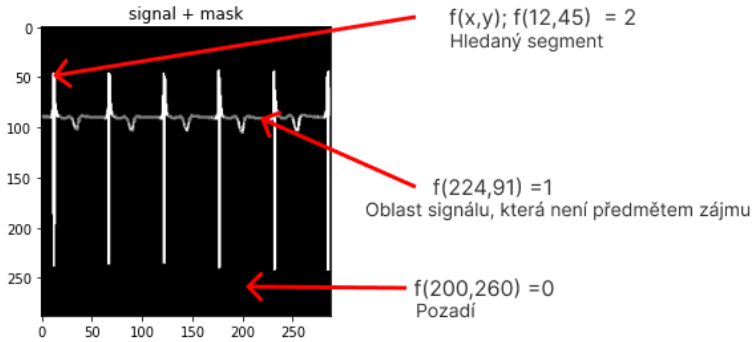
Obrázek 2.20 Predikovaná maska po prahování metodou Otsu

Prahováním bylo dosaženo požadovaného binárního obrazu, nicméně, během prahovaní byly zdůrazněny i pixely mimo požadovanou oblast zájmu. Kompletace byla provedena průnikem obrazu masky a obrazu příslušného signálu. Jelikož byly oba obrazy binární, bylo možné průnik realizovat obyčejným vynásobením obrazu signálu a naprahované masky mezi sebou. Hodnoty, které spolu nesouhlasily, byly přepsány na hodnotu nula. Tímto bylo zajištěno, že v naprahovaném obrazu se budou vyskytovat ty nenulové pixely, které jsou vymezeny signálem. Obraz po průniku je na obrázku **2.21**.



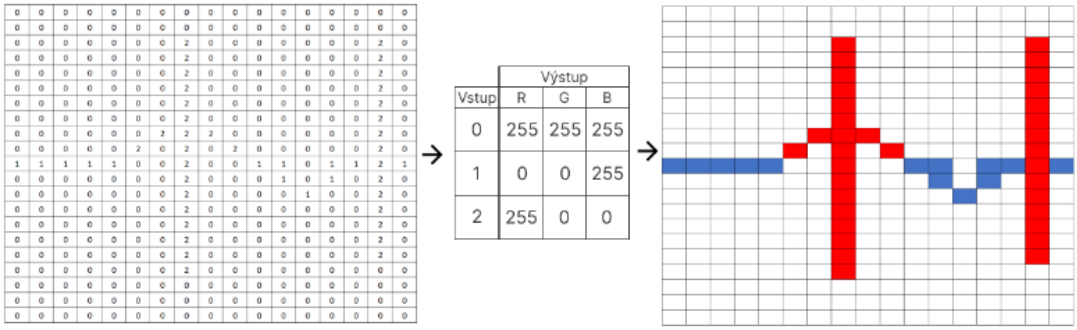
Obrázek 2.21 Vlevo – obraz před průnikem obrazu masky s obrazem signálu, vpravo – výstup průniku

Pro lepší vizuální vjem bylo nakonec provedeno barevné ilustrováni segmentace. Nejdříve se provedl součet obrazu signálu a naprahované (a upravené průnikem) masky. Součtem zůstaly pixely pozadí 0, nicméně, pixely, kde se nacházela hledaná oblast, vymezená maskou, nově nabyla hodnoty 2. Zbylé pixely, tedy pixely signálu, které nebyly součásti hledaného segmentu, měli stále hodnotu 1. Obraz se z binárního proměnil na šedotónový, který ale disponoval pouze třemi hodnotami a to: 0 - pozadí, 1 - signál, 2 - hledaný segment. Tento postup je znázorněn na obrázku **2.22**.

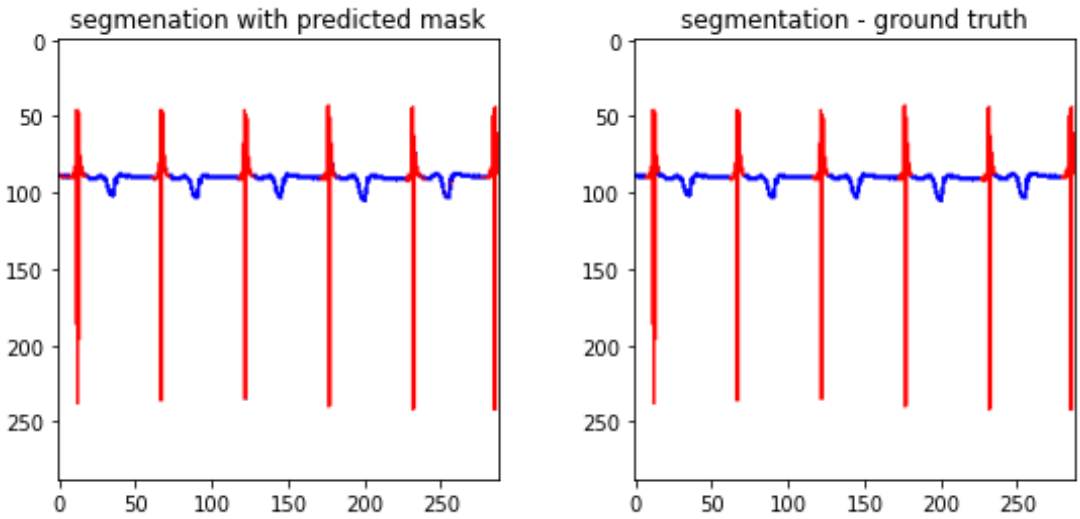


Obrázek 2.22 Příprava obrazu na pseudobarvení

Nakonec se provedlo pseudobarvení, jehož provedení je zjednodušeně ilustrováno na obrázku **2.23**, kdy pixely pozadí se obarvily do bílé barvy, pixely signálu, tedy ty s hodnotou 1, do modré a hledaný segment do barvy červené. Finální výstup segmentace, tedy i po post-processingu, je na obrázku **2.24**.



Obrázek 2.23 Ilustrace řešení provedeného pseudobarvení



Obrázek 2.24 Ilustrace řešení provedeného pseudobarvení

3. ZHODNOCENÍ SEGMENTACE

V případě sémantické segmentace mohou mít jednotlivé pixely obrazu masky pouze jednu ze dvou hodnot: 0 – pozadí a 1 – hledaný segment. Lze tak mluvit o binární proměnné. Lze si tak položit otázku, zda se jedná o pixel masky. Výstupem může být 0/1, ano/ne nebo true/false. Lze tak využít metrik jako například TP (true positive), TN (true negative), FP (false positive) a FN (false negative). [81]

TP v případě této práce říká, kolik pixelů predikovaného obrazu má hodnotu 1 (tedy, že se jedná o pixel masky), je-li u těchto pixelů očekávanou hodnotou1. TN naopak říká, kolik pixelů má hodnotu 0, když mají mít hodnotu 0. Dále pak FP udává počet pixelů, které mají hodnotu 1, mají-li mít hodnotu 0 a FN naopak udává, kolik pixelů má hodnotu 0, mají-li mít hodnotu 1. Tyto metriky lze vložit do takzvané matice záměn (**Tabulka 1**). [81]

Tabulka 1 Matice záměn

		Predikované hodnoty	
		1	0
Skutečné hodnoty	1	TP	FN
	0	FP	TN

Pokud by se pro zhodnocení modelu zohledňovaly pouze TP, TN, FP a FN, mohlo by hodnocení být nepřesné. Hledaná maska by mohla totiž zabírat pouze nepatrný zlomek z celého obrazu, tedy, dominovaly by pixely pozadí (nevyváženost tříd). Hodnota TN by tak byla stále vysoká i když by pixely masky byly predikovány nesprávně. Model by tak mohl být mylně považován za efektivní. Bylo tedy vhodné vyhodnotit model dle jiných metrik. V rámci této práce byly pro zhodnocení modelu využito sensitivity, preciznosti a dice koeficientu. [81]

Sensitivita (také nazývána jako Recall) je formulována jako poměr mezi správně pozitivně klasifikovanými vzorky (TP) a všemi vzorky, které mají být pozitivní (rovnice (3.1)). [74][81][82]

$$senzitivita = \frac{TP}{TP+FN}. \tag{3.1}$$

Preciznost (z anglického Precission) je definována jako poměr mezi správně pozitivně klasifikovanými vzorky a všemi vzorky, které byly klasifikovány jako pozitivní (rovnice (3.2)). [82]

$$preciznost = \frac{TP}{TP+FP}. \tag{3.2}$$

Dice koeficient, známý také jako Sørensen–Dice index, udává, jakou měrou jsou si dvě množiny podobné. Zde jsou množinami myšleny obrazy. Bývá vyjádřen v rozsahu od 0 do 1. Kdy hodnota 1 vyjadřuje nejvyšší podobnost. Je definován rovnicí (3.3) jako poměr mezi dvojnásobkem intersekce a součtu počtu prvků obou množin [83]

$$Dice = \frac{2|X \cap Y|}{|X|+|Y|}, \tag{3.3}$$

využitím TP, FN a FP lze vzorec přepsat do podoby, které je v rovnici (3.4), [83]

$$Dice = \frac{2 \times TP}{2 \times TP + FN + FP}. \tag{3.4}$$

Tyto statistiky jsou důležité pro zhodnocení získaných výstupů, které budou prezentovány v následujícím textu. Nejdříve budou zobrazeny získané výsledky z navrhnutých architektur (řešení diplomové práce), následně bude provedeno porovnání s frameworkem třetí strany Segmentation Models.

3.1 Získané výstupy z navrhnutých architektur

V této práci bylo dosaženo následujících výsledků. Jedná se o výstupy, které vznikly predikcí masky ze vstupního testovacího obrazu, následnou úpravou této masky a kompletací segmentace. Vymyšlený proces byl automatický, kromě zapsání cesty k testovacímu obrazu nebyl v rámci segmentace proveden žádný ruční zásah. Nejdříve budou zobrazeny výstupy z U-net a následně, pro porovnání, výstupy ze SegNet.

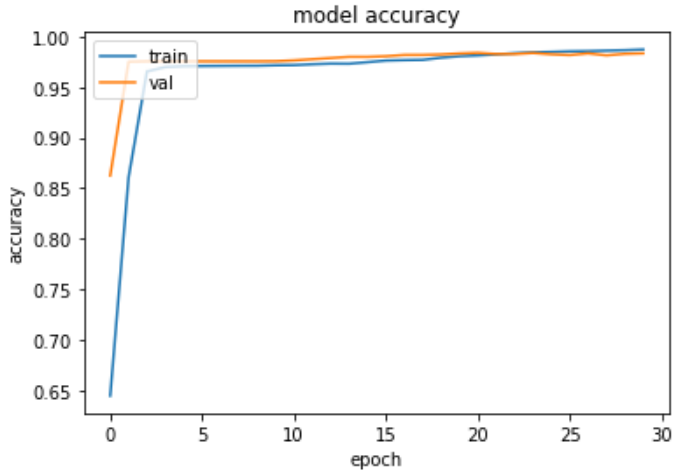
Pro obě architektury byla použita stejná množina dat. **Trénovací množina dat** disponovala **118** obrazy signálů (a tím pádem 118 obrazů příslušných masek) a **testovací množina 7 obrazy** signálů (a tím pádem 7 obrazy příslušných masek). Po naučení byly otestovány všechny testovací obrazy, jejich výstupy jsou zobrazeny níže. Zmíněné včetně dalších doplňujících informací je v **Tabulka 2**.

Tabulka 2 Počet parametrů obou architektur, počet epoch učení a velikost množin dat. Vysvětlivky tabulky: 1 – parametry (váhy), které se zpětným šířením chyby upravují, 2 – parametry, které se zpětným šířením chyby neupravují; 3 – trénovací množina dat obrazů signálů; 4 – testovací množina dat obrazů signálů.

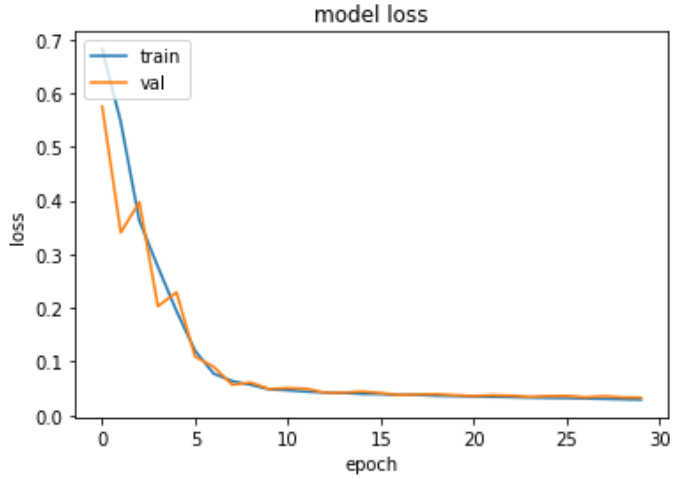
	U-net	SegNet
Celkový počet parametrů	1 940 917	1 770 337
Trénovatelné parametry[1]	1 940 917	1 768 417
Netrénovatelné parametry[2]	0	1920
Počet epoch	30	50 (učení ukončeno u 40. epochy)
Trénovací dataset[3]	118	118
Testovací dataset[4]	7	7

3.1.1 U-net

V následující sekci jsou výstupy získané pomocí navrhnutého U-net. Z důvodu většího množství výstupních dat budou ilustrovány pouze vybrané výstupy segmentace ze čtyř testovacích obrazů. Všechny výstupy jsou k nahlédnutí v příloze. Model se učil 30 epoch (trénování nebylo předčasně ukončeno). Kdy metriky poslední epochy byly následující: trénovací chyba **0.0293**, trénovací přesnost **0.987**, validační chyba: **0.0334** a přesnost validační přesnost **0.984**. Průběh vývoje přesnosti a chyby v epochách je na obrázku **3.1** a **3.2**. Model byl otestován na 7 testovacích obrazech. Zhodnocení je v **Tabulka 3**



Obrázek 3.1 Průběh vývoje trénovací (modrá křivka) a validační (oranžová křivka) přesnosti v epochách u U-net

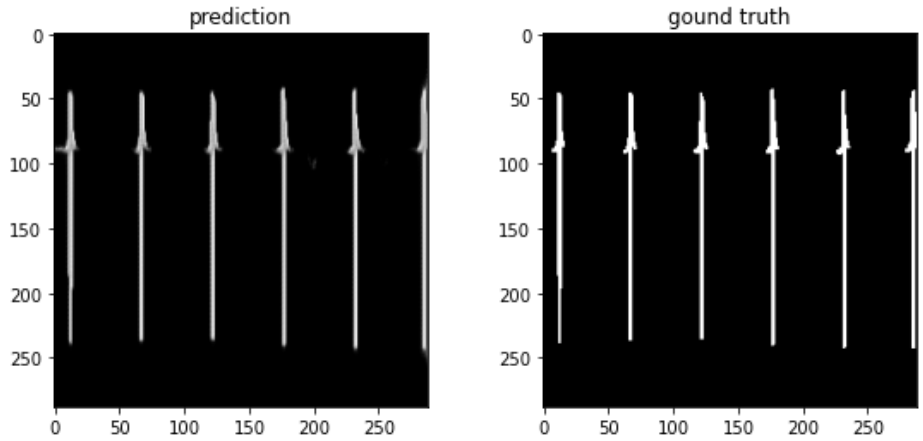


Obrázek 3.2 Průběh vývoje trénovací (modrá křivka) a validační (oranžová křivka) chyby v epochách u U-net

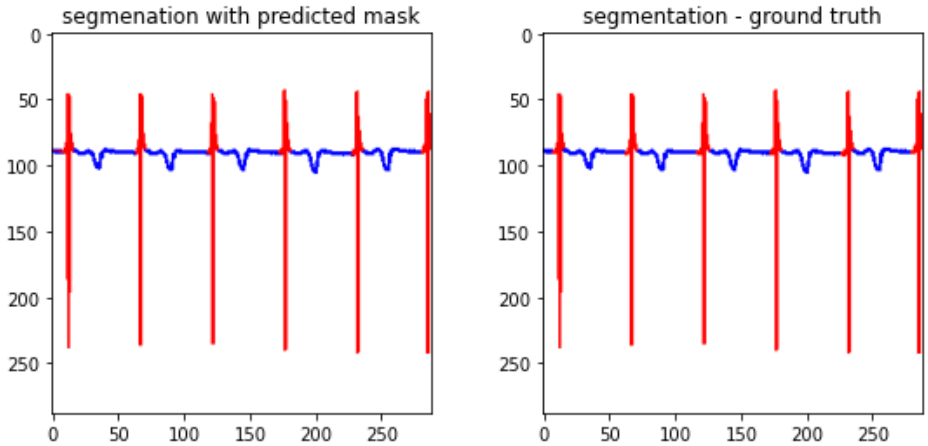
Tabulka 3 Zhodnocení segmentace pomocí navrhnutého U-net na testovacích datech

Obraz č.	TP	TN	FP	FN	Preciznost	Sensitivita	Dice
1	4034	78830	46	31	0.989	0.992	0.990
2	3832	77323	1761	28	0.685	0.992	0.811
3	3574	78846	439	85	0.891	0.977	0.932
4	3569	79223	56	96	0.985	0.974	0.979
5	4296	78266	294	88	0.940	0.980	0.957
6	4365	77955	534	90	0.891	0.980	0.933
7	4005	77642	1207	90	0.768	0.978	0.861

U-net: Testovací obraz č.1

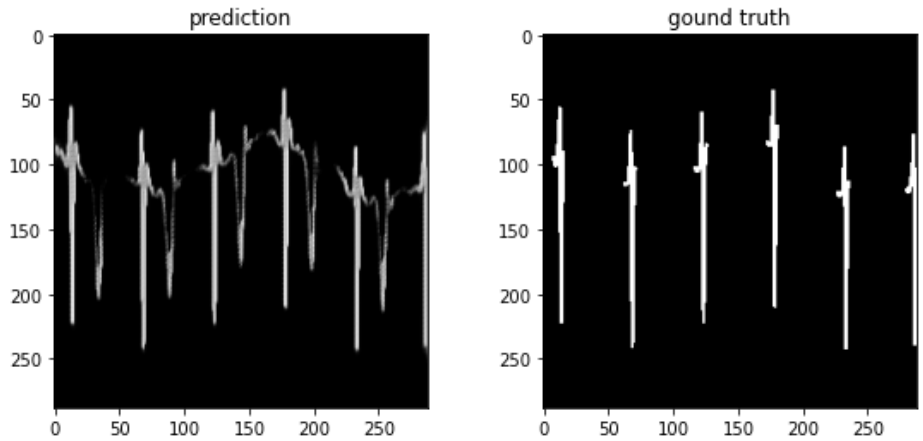


Obrázek 3.3 Vlevo – predikce masky pomocí U-net, vpravo – požadovaný výstup

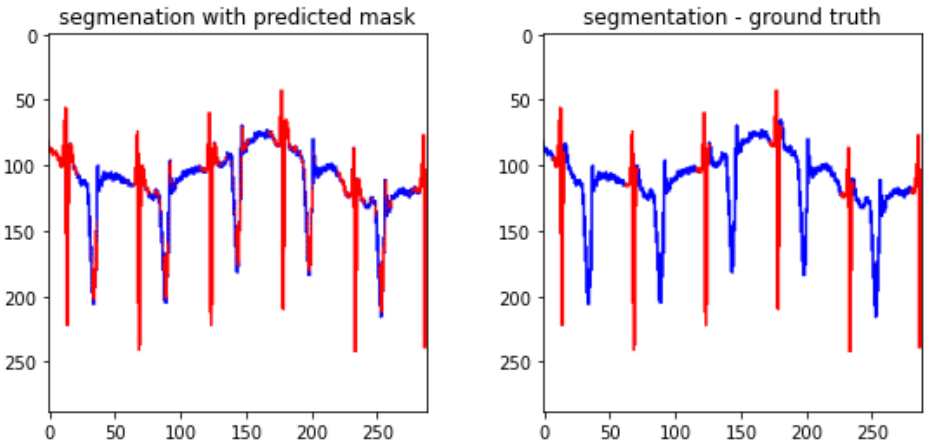


Obrázek 3.4 Vlevo – výstup segmentace za pomoci U-net (dice 0.990), vpravo – požadovaný výstup

U-net: Testovací obraz č.2

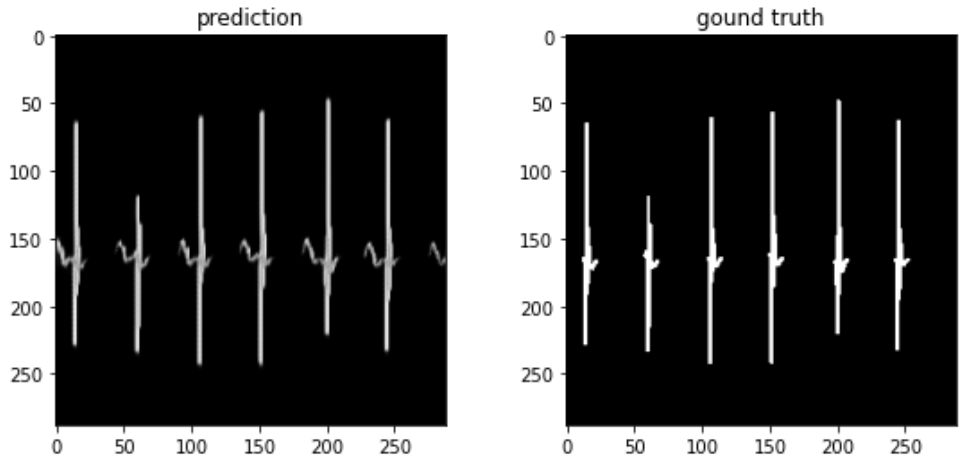


Obrázek 3.5 Vlevo – predikce masky pomocí U-net, vpravo – požadovaný výstup

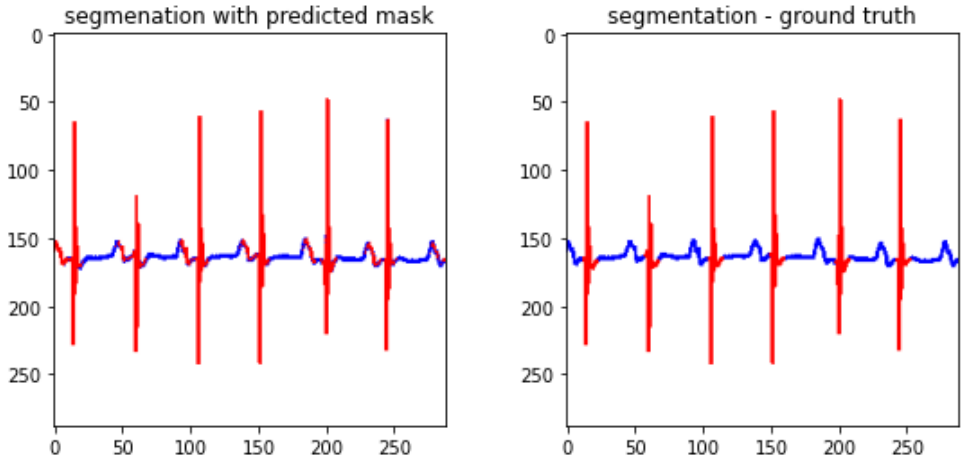


Obrázek 3.6 Vlevo – výstup segmentace za pomoci U-net (dice 0.811), vpravo – požadovaný výstup

U-net: Testovací obraz č.3

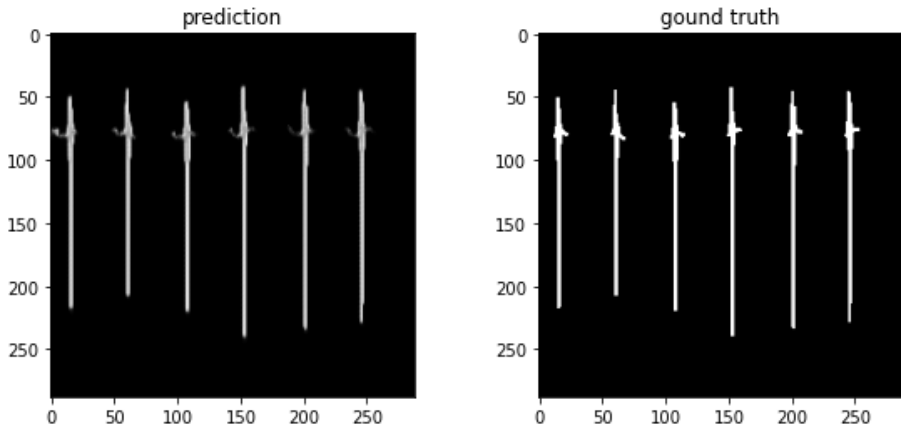


Obrázek 3.7 Vlevo – predikce masky pomocí U-net, vpravo – požadovaný výstup

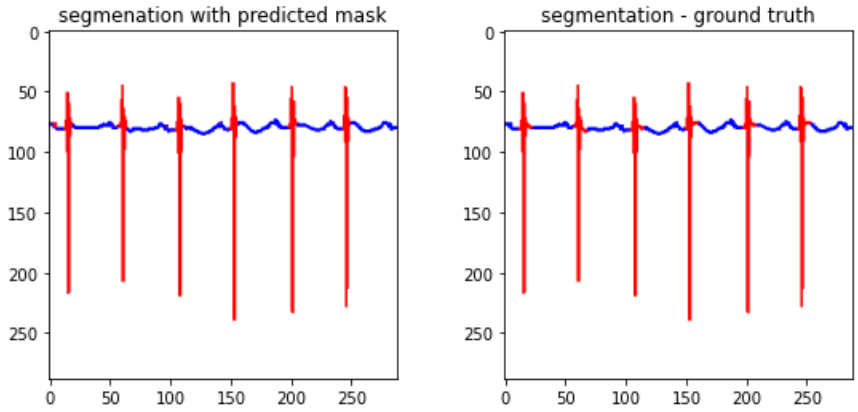


Obrázek 3.8 Vlevo – výstup segmentace za pomoci U-net (dice 0.932), vpravo – požadovaný výstup

U-net: Testovací obraz č.4



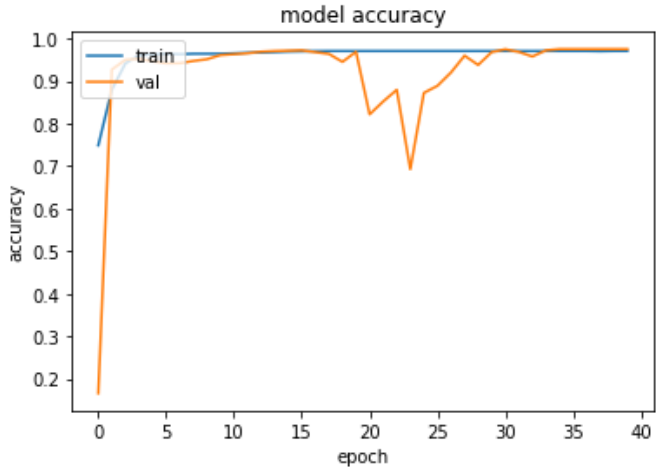
Obrázek 3.9 Vpravo – výstup segmentace za pomoci U-net, vpravo – požadovaný výstup



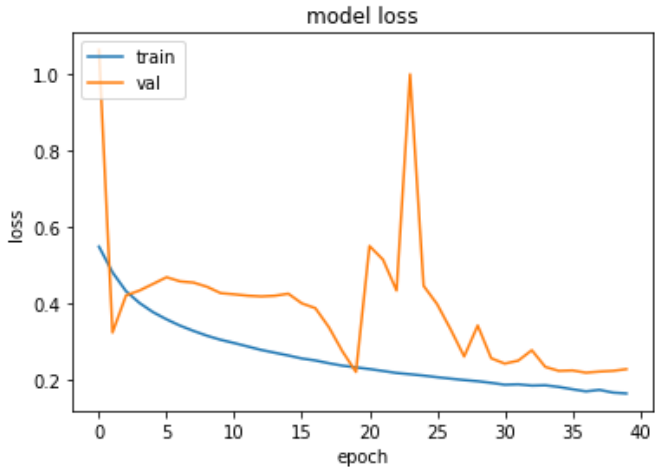
Obrázek 3.10 Vlevo – predikce masky pomocí U-net (dice 0.979), vpravo – požadovaný výstup

3.1.1 SegNet

V následující sekci jsou výstupy získané pomocí navrhnutého SegNet. Z důvodu většího množství výstupních dat budou ilustrovány pouze vybrané výstupy ze čtyř testovacích obrazů (zbylé výstupy se nachází v příloze). U modelu bylo zvoleno 50 epoch. Předčasné ukončení bylo dle validační chyby (stejně jako u U-net). Pokud se validační chyba nezlepšila ani po 10 epochách (u U-net bylo nastaveno po 5 epochách), model se ukončil. U SegNet bylo učení ukončeno u 40. epochy. Kdy metriky poslední epochy byly následující: trénovací chyba 0.1661, trénovací přesnost 0.9706, validační chyba: 0.2295 a přesnost validační přesnost 0.975. Průběh vývoje přesnosti a chyby v epochách je na obrázku **3.11** a obrázku **3.12**. Model byl otestován na 7 testovacích obrazech. Zhodnocení segmentace je v **Tabulka 4.**



Obrázek 3.11 Průběh vývoje trénovací (modrá křivka) a validační (oranžová křivka) přesnosti v epochách u SegNet

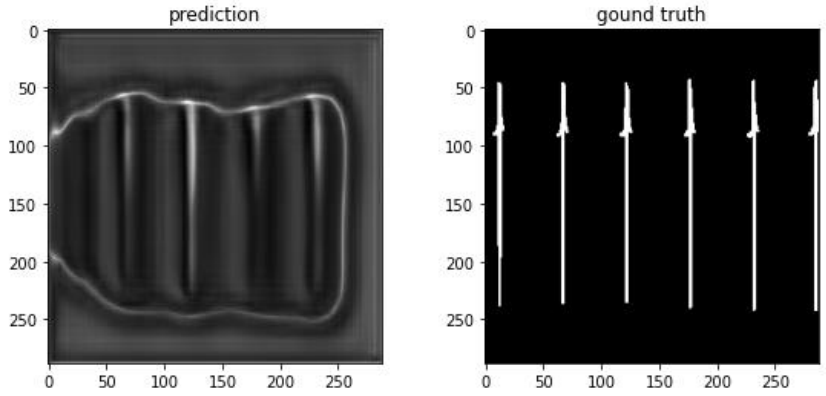


Obrázek 3.12 Průběh vývoje trénovací (modrá křivka) a validační (oranžová křivka) chyby v epochách u SegNet

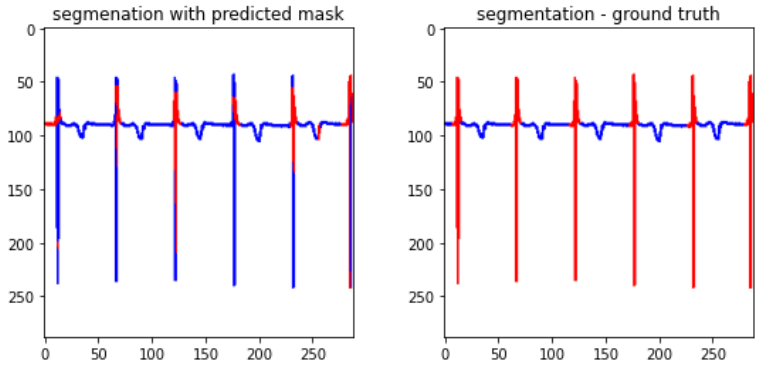
Tabulka 4 Zhodnocení segmentace pomocí navrhnutého SegNet na testovacích datech

Obraz č.	TP	TN	FP	FN	Preciznost	Sensitivita	Dice
1	1403	70242	8634	2665	0.140	0.345	0.199
2	1661	66849	12235	2199	0.120	0.430	0.187
3	2014	50790	28495	1645	0.066	0.550	0.118
4	1667	73484	5795	1998	0.223	0.455	0.300
5	1915	72840	5720	2469	0.251	0.437	0.320
6	2293	53796	24693	2162	0.085	0.515	0.146
7	1880	67974	10875	2215	0.147	0.459	0.223

SegNet: Testovací obraz č.1

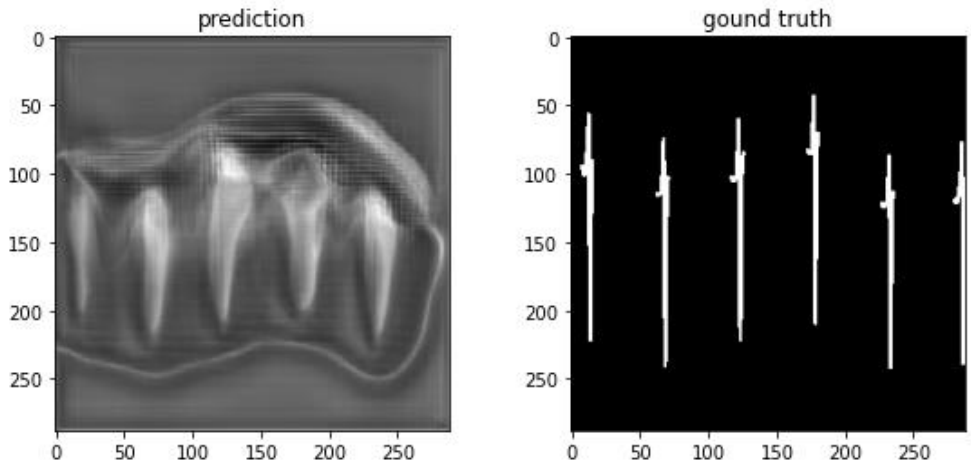


Obrázek 3.13 Vlevo – výstup segmentace za pomoci SegNet, vpravo – požadovaný výstup

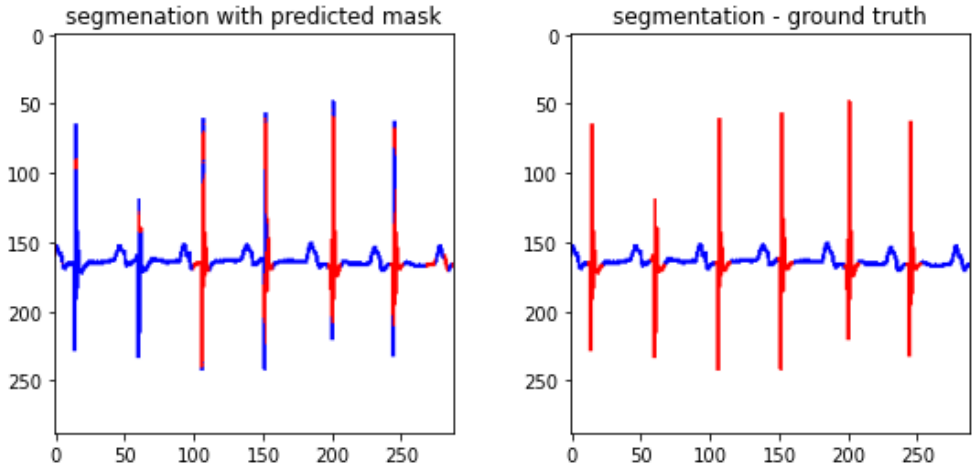


Obrázek 3.14 Vlevo – predikce masky pomocí SegNet (dice 0.199), vpravo – požadovaný výstup

SegNet: Testovací obraz č.2

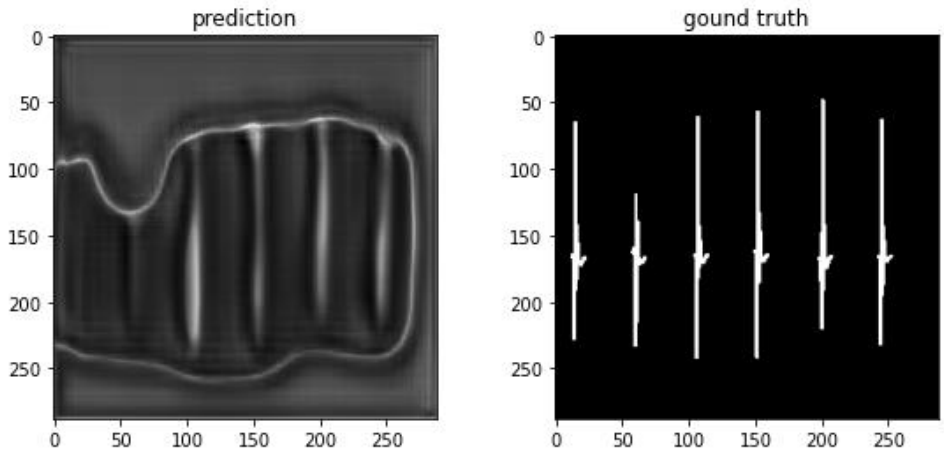


Obrázek 3.15 Vlevo – výstup segmentace za pomoci SegNet, vpravo – požadovaný výstup

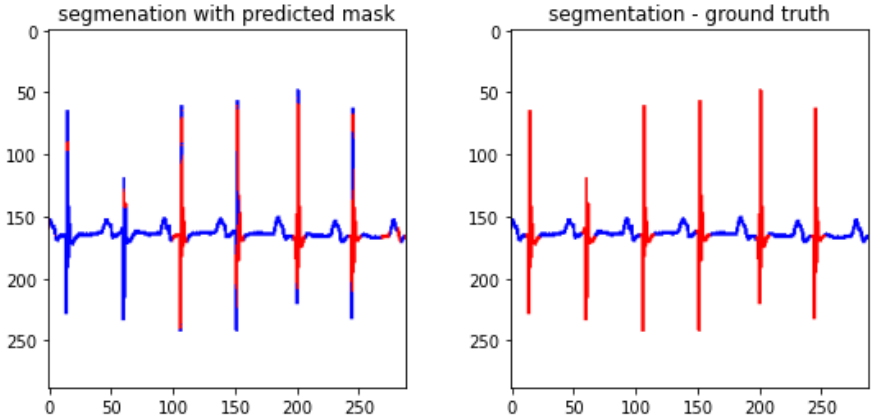


Obrázek 3.16 Vlevo – predikce masky pomocí SegNet (dice 0.187), vpravo – požadovaný výstup

SegNet: Testovací obraz č.3

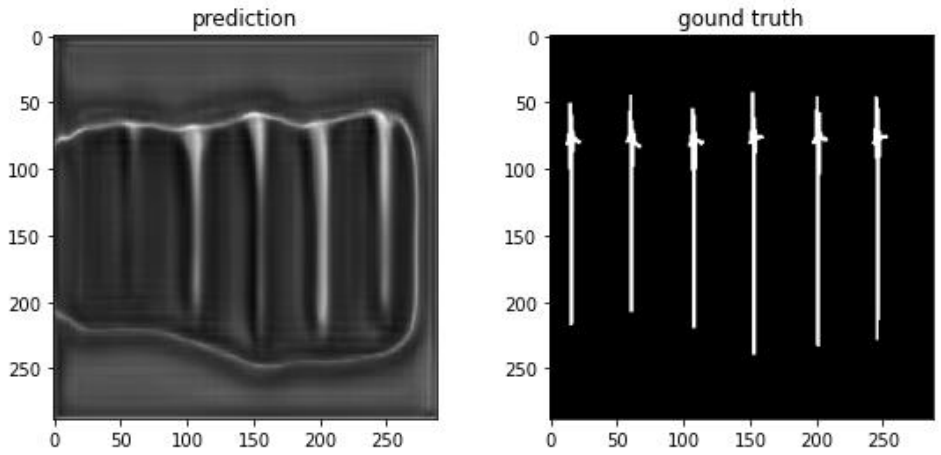


Obrázek 3.17 Vlevo – výstup segmentace za pomoci SegNet, vpravo – požadovaný výstup

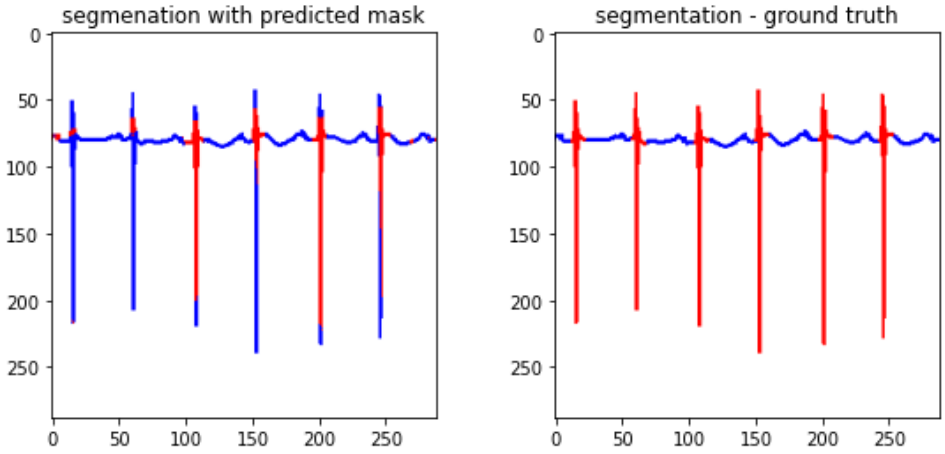


Obrázek 3.18 Vlevo – predikce masky pomocí SegNet (dice 0.118), vpravo – požadovaný výstup

SegNet: Testovací obraz č.4



Obrázek 3.19 Vlevo – výstup segmentace za pomoci SegNet, vpravo – požadovaný výstup



Obrázek 3.20 Vlevo – predikce masky pomocí SegNet (dice 0.300), vpravo – požadovaný výstup

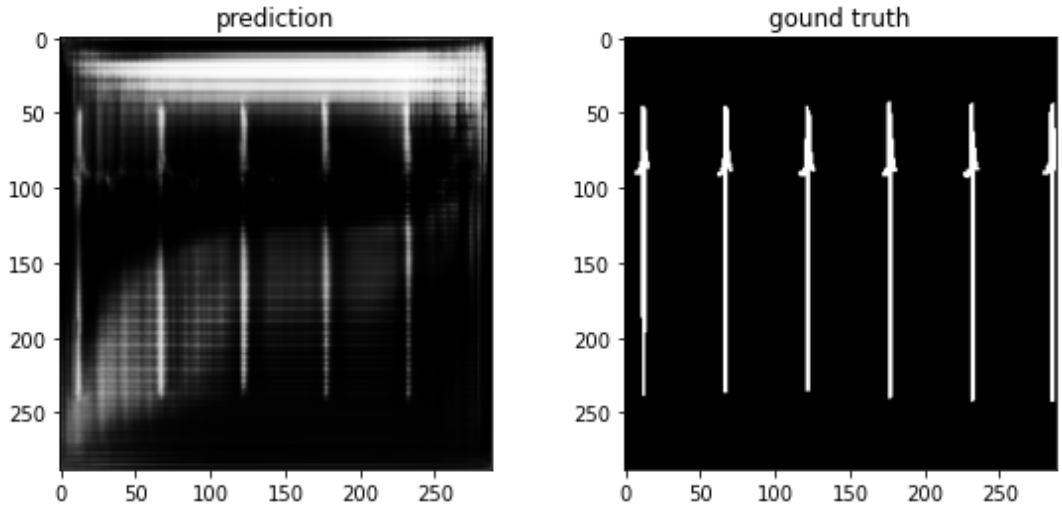
3.2 Porovnání segmentace s řešením z třetí strany

Pro porovnání byl výstup ze segmentace porovnán s frameworkem třetí strany, Segmentation Models, který je volně ke stažení na GitHub https://github.com/qubvel/segmentation_models. Jedná se o framework s umělými neuronovými sítěmi pro segmentaci obrazů, také založených na knihovně TensorFlow.

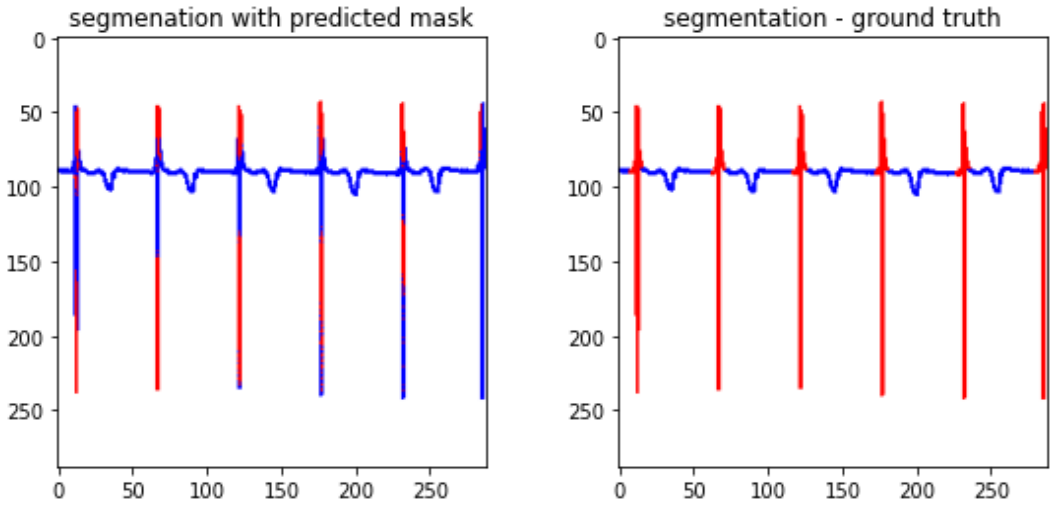
Disponuje čtyřmi modely a to U-net, Linknet, PSPNET, FNP. Byla použita stejná data jako u vytvořeného modelu. Pro porovnání se použil U-net.

Model se definoval přes `sm.Unet()`, kde vstupem byl typ Backbone, tedy extraktoru příznaků (zadával se jako textový řetězec), který byl zde *resnet-34*. Framework umožnuje také vložit jako další parametr v metodě `Unet()` již natrénované váhy. Do druhého vstupu se tak mohlo vložit `encoder_weights='imagenet'` nebo se místo `'imagenet'` dalo `None`, čímž se vygenerovaly náhodné váhy. Zvolily se náhodně vygenerované váhy. Nakonec se definoval optimalizační algoritmus a ztrátová funkce modelu. Jelikož je tento framework také založen na Keras TensorFlow, byla použita stejná funkce `compile()`. Zde se jako kriteriální funkce opět zvolila binární křížová entropie a Adam.

Dále následovalo trénování a predikce. Predikovaná maska vytvořeného řešení U-net a výstupu U-net ze Semgmentations Models je porovnán na obrázku **3.21**, výsledná segmentace pak na obrázku **3.22**.



Obrázek 3.21 Vlevo – predikce ze Segmentation Models, vpravo – ground truht



Obrázek 3.22 Vlevo – výsledná segmentace testovacího obrazu č.1 za využití U-net ze Segmentation Models (0.205), vpravo – výstup z navrhnutého U-net

3.3 Porovnání řešení se zahraničními studiemi na podobné téma

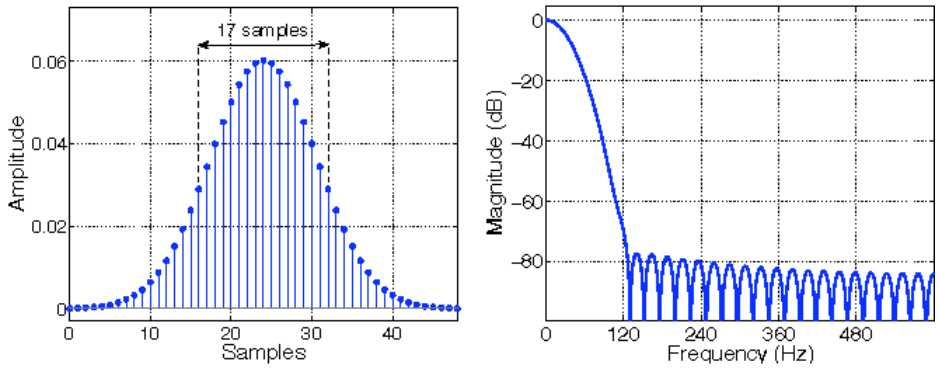
Přístup, který byl v této práci zvolen, lze považovat za méně obvyklý, proto se nepodařilo nalézt studii, která by prováděla segmentaci intrakardiálního záznamu stejným způsobem jako v této práci. Byly však nalezeny studie, jejichž témata byla do jisté míry podobná.

Například článek **A new approach for automated location of active segments in intracardiac electrograms** z Karlsruhské univerzity se věnoval segmetnaci intrakardiálního záznamů v 1D oblasti pomocí upraveného Pan-Tomkinsonova algoritmu. [84]

Pan-Tomkins je real-time algoritmus detekující komplexy QRS, který vyvinuli Jiapu Pan a Willis J. Tomkins. Je provedeno předzpracování signálu, z důvodu utlumení šumu (a zvýraznění komplexů QRS), pásmovou propustí, která je tvořena kaskádou horní a dolní propusti. Následuje diferenciace pro nalezení velkého sklonu, umocnění vzorků signálu na druhou (zvýrazní se vyšší frekvence a zabrání se tak falešně pozitivním detekcím způsobených detekcí T vln), integrace pohyblivým oknem (pro extrakci příznaků jako sklon a šířka QRS), prahování (na základě nějž se určí, zda se jedná o QRS či nikoliv) a v případě potřeby dodatečné úpravy (například pro zobrazení výstupu). [85]

V práci z Karlsruhské univerzity byl Pan-Tompkinsnuv algoritmus upraven pro potřeby řešení jejich problému. Stejně u původního Pan-Tompkinsonova algoritmu i zde bylo provedeno předzpracování (filtrace, případně i eliminace elevaci izlolínie), nicméně další kroky už byly už odlišné. Po předzpracování následovalo vymezaní sekcí s vysokou frekvencí a amplitudou pomocí pro odlišení „neaktivních oblastí“ (oblast izolinie) od oblastí aktivních. To bylo dosaženo pomocí Teagerova nelineárního energetického

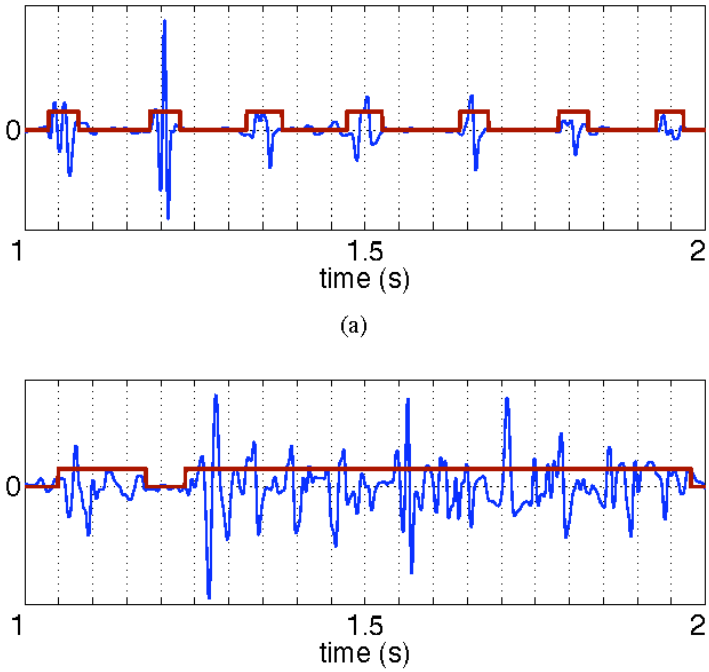
operátoru (NLEO). Dalším krokem byla provedena filtrace výstupu NLEO pomocí Gaussovy dolní propusti [84][85]. Impulzní a frekvenční charakteristika této dolní propusti je na obrázku **3.23**.



(a) Impulse response (b) Frequency response

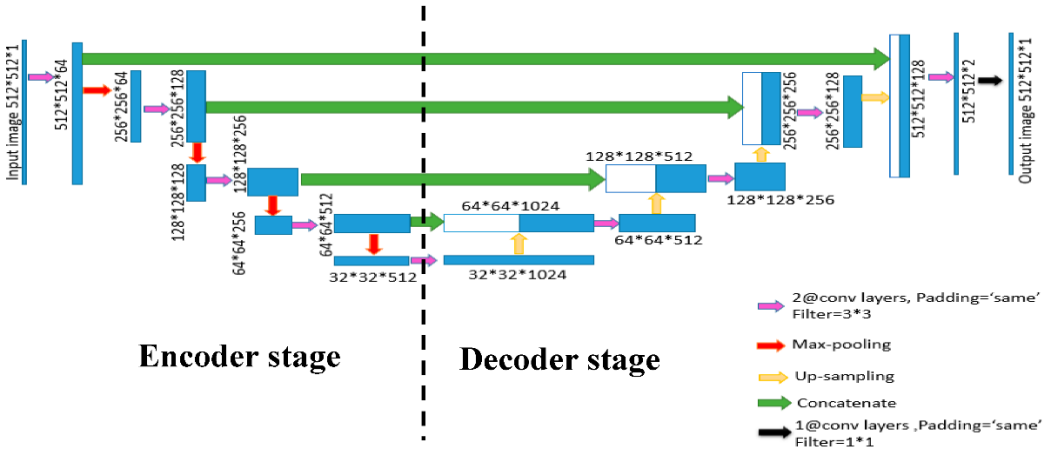
Obrázek 3.23 Impulzní (a) a frekvenční charakteristika (b) Gaussovy dolní propusti

Nakonec bylo provedeno prahování a případné dodatečné úpravy. Výsledná segmetnace je na obrázku **3.24**. Z výstupu si lze povšimnout, že v případě pravidelně se opakující aktivity model jednotlivé úseky detekuje. Segmentované úseky oproti řešení této diplomové práce jsou vyznačeny křivkou, která „obaluje“ hledané segmenty. Další odlišností je, že jsou hledané všechny aktivní úseky, příklad v (b), kdy je kontinuální elektrická aktivita, je vybrán celý úsek, nehledě na rozlišení síňové či komorové aktivity. [84]



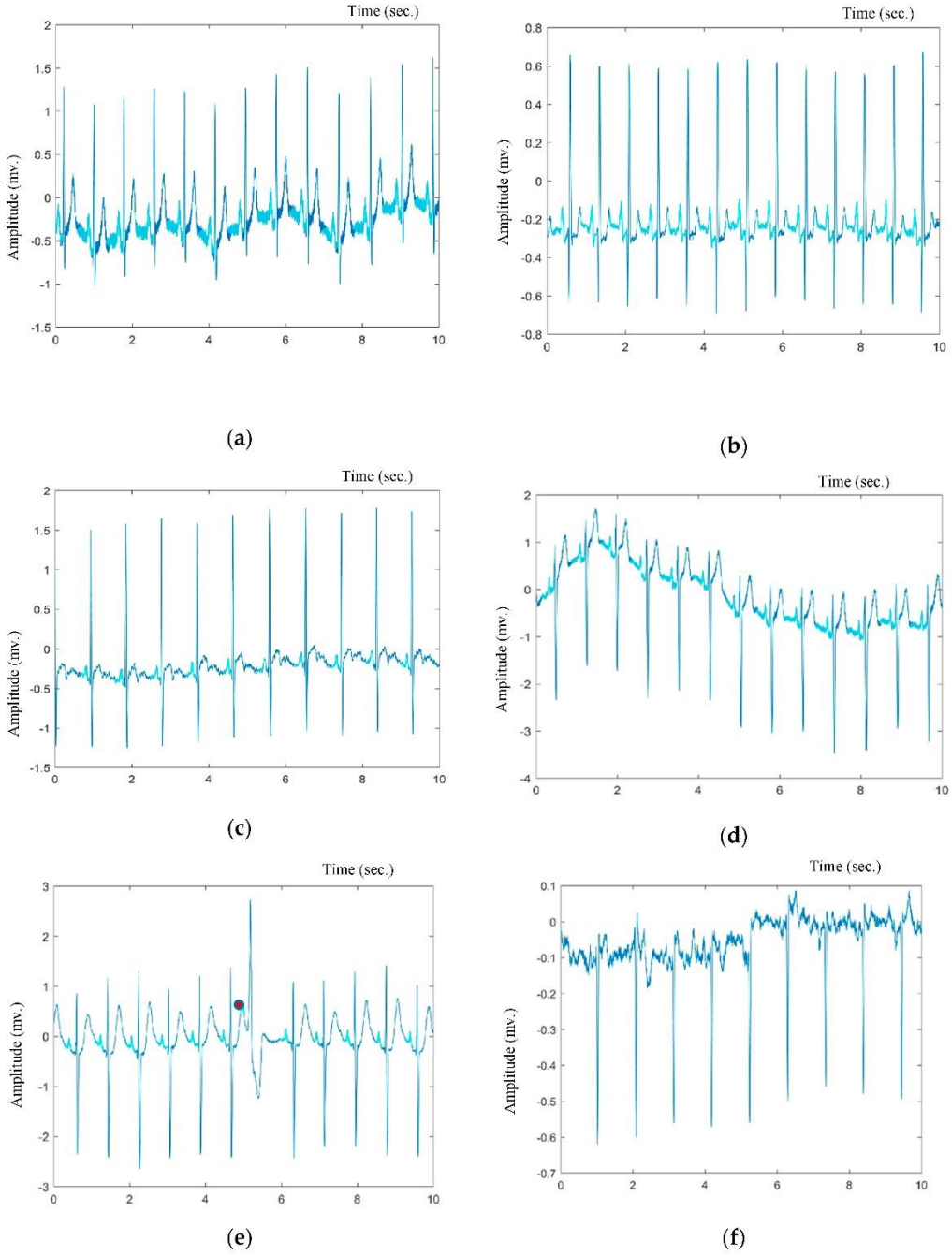
Obrázek 3.24 Výsledek segmentace intrakardiálního záznamu: (a) s proměnlivou amplitudou, (b) během kontinuální elelektrické aktivity

Další článek **P-Wave Detection Using a Fully Convolutional Neural Network in Electrocardiogram Images** z Arabské Akademie pro vědu, technologie a námořní transport (Káhira), se sice věnuje detekci P vln povrchového EKG, ale byl realizován pomocí konvoluční neuronové sítě U-net ve 2D oblasti. Základní kroky realizace byly podobné v praktické části. Nejdříve se provedla ruční anotace EKG signálů a následně se vytvořila architektura U-net, která je na obrázku **3.25**. [74]



Obrázek 3.25 U-net použitý v řešení

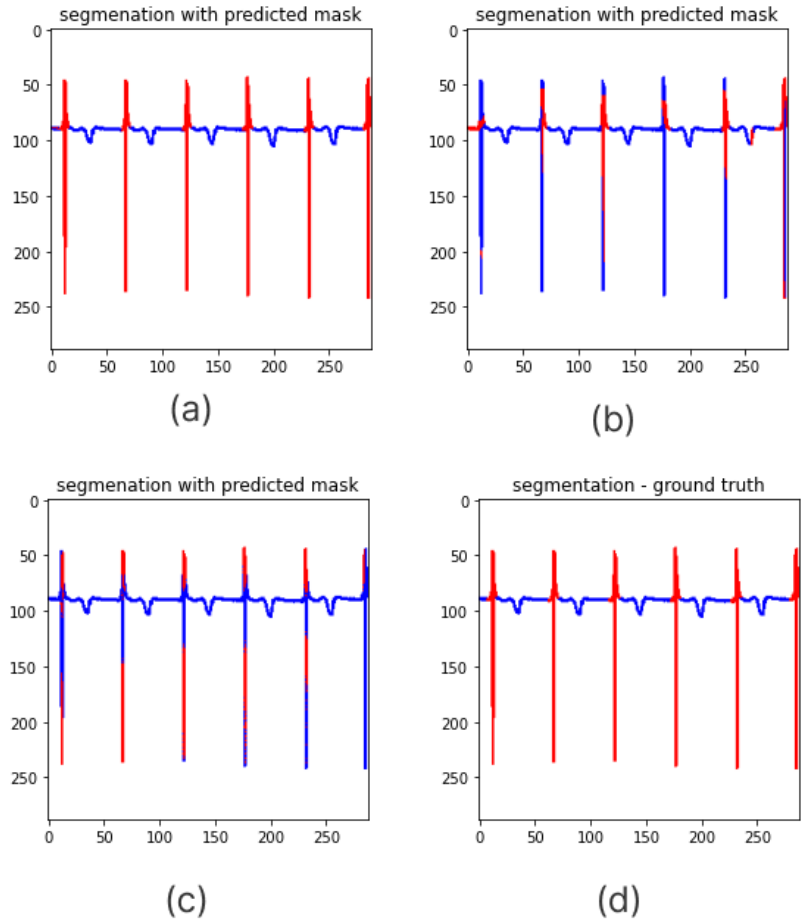
Rozdíl oproti řešení diplomové práce byla podoba v množiny dat. Zatímco v diplomové práci byly obrazy signálu binární, zde byly obrazy signálu barevné. Obrazy masky byly nicméně také binární. Také byl odlišný počet trénovacích dat, kdy řešení z článku disponovalo výrazně větší množinou dat. Výstupy jsou na obrázku **3.26**. [74]



Obrázek 3.26 Detekované vlny při: (a) vysokofrekvenčnímu šumu, (b) fyziologickém rytmu bez U vln, (c) fyziologickém rytmu s U vlnami, (d) fyziologickém rytmu s negativní polarizací, (e) spojení P a T vlny, (f) žádná detekce při atrialní fibrilaci

4. DISKUZE

Ze získaných výsledků je patrné, že navrhnutá alternativa architektury **U-net** dosáhla nejuspokojivějších výsledků oproti navrhnuté architektuře **SegNet** a **U-net** z frameworku Segmentations Models. U navrhnuté U-net měla nejzdařenější segmentace **dice kolem 0.99**, zatímco u SegNet byl **dice 0.318** a u U-net ze Segmentation Models **pouze 0.205**. Například pro testovací obrázek č.1 ilustrováno porovnání na obrázku **4.1**.

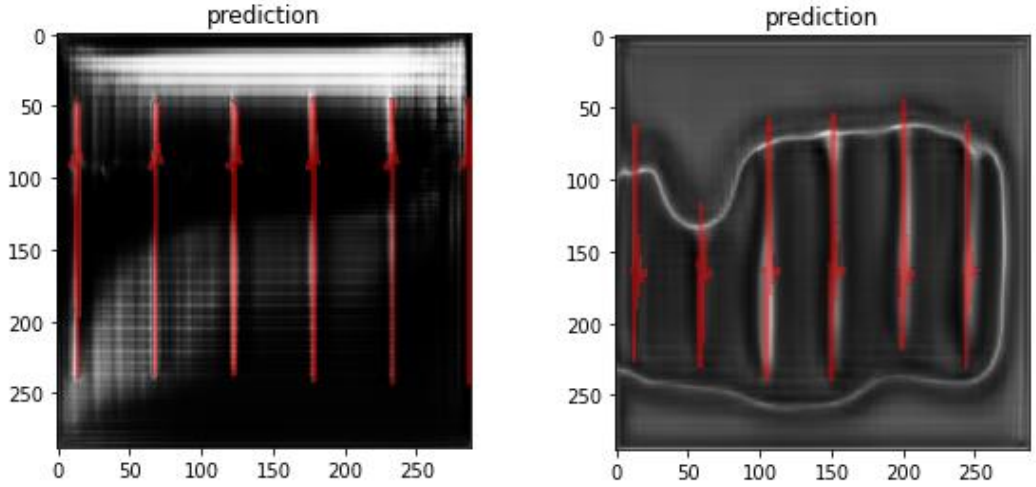


Obrázek 4.1 Segmentace testovacího obrazu č.1 za využití: (a) navrhnuté alternativy U-net (dice segmentovaného obrazu - 0.99), (b) navrhnuté alternativy SegNet (dice segmentovaného obrazu - 0.199), (c) U-net ze Segmentation Models (dice segmentovaného obrazu - 0.205); (d) požadovaný výstup

Nutno zmínit, že například architektura SegNet disponuje menším počtem parametrů a podoba architektury byla oproti SegNet z teoretického úvodu (obrázek **2.13**) upravena výrazněji. Například ve výstupní vrstvě bylo místo aktivační funkce softmax zvolena sigmoida. Také co se týče Segmentation Models, nelze na základě získaných výsledků dělat závěr, že tento framework je neefektivní. Je totiž určen pro barevné a šedotónové

obrazy, které jsou spíše spjaté s objekty reálného světa než s převedenými obrazy signálu a masek do obrazů.

I přesto, že SegNet a U-net ze Segmentation Models nebyly tak zdárné, oproti navrhnutému U-net, byly u predikovaných masek pozorovány náznaky výskytu správných segmentovaných úseků (obrázek **4.2**). Je možné, že jinými (vhodnějšími) dodatečnými úpravami by mohly výsledné segmentace vypadat mnohem lépe. Za zmínění ale stojí skutečnost, že když byl u SegNet zvolen stejný počet epoch jako u U-net, byly výsledky horší, než jaké jsou v této práci ilustrovány. Proto bylo u SegNet zvoleno 50 epoch učení, přičemž díky `EarlyStopping` (a nastavením argumentů) se proces učení ukončil už u 40. epochy. I přesto bylo potřeba více epoch učení.



Obrázek 4.2 Náznaky výskytu segmentů u predikovaných obrazů masek za využití (vlevo) U-net ze Segmentation Models, (vpravo) SegNet, které by mohly odpovídat požadované podobě masky (červeně zvýrazněné)

Ačkoli byly výstupy z **navrhnutého U-net** úspěšnější než například výstupy ze SegNet, byly u většiny testovacích obrazů ve vetší/menší míře FP pixely. V případě nehledaných segmentů, které byly vodorovné, bez výrazných záchvěvů (zvlnění křivky), byly správně považovány za nehledané oblasti (TN). Nicméně, pokud byl u nich výraznější záchvěv (oblast komorové aktivity) došlo k mylné klasifikaci (obrázek 3.6), kdy tyto pixely byly zařazeny do masek (FP). Možným důvodem mohla být skutečnost, že síť mohla rozlišovat jednotlivé segmenty dle vertikálních hran, hrany těchto záchvěvů mívaly podobný směr jako hrany oblastí, reprezentující masky.

Důležitým faktorem úspěšnosti byla nejen zvolená architektura, ale i data, které byly síti poskytnuty. Pokud by procesem vytváření obrazů nesouhlasily prostorové souřadnice bílých pixelů obrazů masek s pixely příslušných obrazů signálu, segmentace by se neprovedla (problém ilustruje obrázek **2.7**).

Další důležitou informací je skutečnost, že segmentace neprobíhala nikoliv v 1D oblasti, ale ve 2D. Při práci s obrazy se pracovalo byly extrahovány příznaky jako

například rohy. Časová informace signálu tak zanikla, a tak nelze s výstupy segmentace již dále pracovat v časové oblasti. Nedostatek by však šlo částečně vyřešit tak, že by během vytváření obrazu signálu a masek se zároveň vytvořil obraz os signálu (případně se mohly uložit data o signálu například do JSON či XML). Po dokončené segmentaci by se pak tyto informace mohly využít.

O úspešnosti také rozhodovalo, jak se zrovna model naučil. Nastaly případy, kdy predikovaná maska byla velmi podobná požadované podobě masky. V jiném případě, když se provedlo naučení sítě znovu (vytvořily se tedy i nové parametry, které přeuložily stará .h5 soubor), docházelo někdy k nezdárným predikcím. Spíše, než o chybu se zřejmě jedná o vlastnost, se kterou je potřeba u konvolučních neuronových sítí počítat. Tedy, je potřeba počítat se skutečností, že síť někdy naučí lépe a někdy naopak hůře.

Dále bylo navrhnuté a realizované řešení diplomové práce porovnáno se zahraničními studiemi, které se věnovaly podobnému tématu. První řešení se věnovalo segmetnaci intrakardiálních záznamů v 1D za využití Pan-Tompkinsonova detektoru. Výsledky byly velmi zdárné, nicméně, oproti této práci se hledaly pouze aktivní oblasti a nerozlišovalo se, zda se jedná o aktivitu síní či komor (byly tak hledány všechny úseky mimo izolinií).

Druhá studie prováděla detekci P vln povrchového EKG za využití U-net. Výstupy byly velmi zdárné, dokonce některé obrazy signálů, kde byla přítomna elevace izolinie, byly segmentovány lépe. Je nicméně na místě zohlednit skutečnost, že zmíněné řešení disponovalo jinou množinou dat a také do U-net vstupovaly větší obrazy a také podoba sítě byla mírně odlišná. Nabízí se tedy otázka, zda zmíněné nebylo jedním z faktorů vyšší úspěšnosti.

5. ZÁVĚR

Tato diplomová práce se věnovala segmentaci intrakardiálních záznamů. První část tvořilo teoretické seznámení problematiky, například, co je EKG, rozdíl mezi povrchovým a intrakardiálním EKG a také byla věnována pozornost například konvoluční neuronové síti.

V praktické části byla nejdříve provedena ruční anotace iEKG signálu v softwarovém nástroji SignalPlant. Následovala realizace modelu automaticky segmentující iEKG signál. Předtím však bylo nutné vhodně předzpracovat data, aby je model byl schopný zpracovat. To bylo provedeno ve vývojovém prostředí Spyder, pomocí programovacího jazyka Python. Segmentace neprobíhala přímo v 1D oblasti, ale ve 2D. Byly zvoleny dvě neuronové sítě a to U-net a SegNet, které ale byly v této práci upraveny do jiné podoby. Pro porovnání byla také použita knihovna U-net z frameworku Segmentation Models s podobnou konfigurací jako u navrhnutého U-net (stejná kriteriální funkce a optimalizační algoritmus).

Na základě výsledku bylo patrné, že výstupy z navrhnutého U-net byly mnohem uspokojivější než u SegNet a již hotového U-net ze Segmentation Models.

Nakonec bylo provedeno srovnání se zahraničními studiemi na podobné téma. Řešení bylo porovnáno se dvěma studiemi. První se věnovalo segmentaci aktivních oblastí intrakardiálních záznamů. Oproti řešení této diplomové práce byla segmentace prováděna ve 1D a nikoliv pomocí konvoluční neuronové sítě, ale pomocí upraveného Pan-Tompkins operátoru. Výstupní segmentace nerozlišovala síňovou a komorou aktivitu, pouze rozlišovala aktivní a neaktivní úseky (izolinie). Navíc byly hledané úseky zvýrazněny pouze pomocí křivky, která tyto oblasti „obalovala“. Další studie, se kterou byla diplomová práce porovnána [74] se věnovala detekci P vln povrchového EKG, pomocí 2D konvoluční neuronové sítě (U-net). Výstupy této segmentace byly z hlediska vizuální podoby i statistiky výstupů zdárnější.

LITERATURA

[1] Willem Einthoven. *Engineering and Technology History Wiki* [online]. [cit. 2021-03-18].

[2] S. SALADIN, Kenneth. *Anatomy & Physiology: The Unity of Form and Function*. 9th Ed. Dubuque, IA: McGraw Hill, 2020. ISBN 978-1260256000.

[3] ČIHÁK, Radomír. *Anatomie*. 2., upr. a dopl. vyd. Ilustroval Milan MED, ilustroval Ivan HELEKAL. Praha: Grada, 2004. ISBN 80-247-1132-x.

[4] HUDÁK, Radovan a David KACHLÍK. *Memorix anatomie*. 4. vydání. Ilustroval Jan BALKO, ilustroval Šárka ZAVÁZALOVÁ. Praha: Triton, 2017. ISBN 978-80-7553-420-0.

[5] PETŘEK, Josef. *Základy fyziologie člověka pro nelékařské zdravotnické obory*. Praha: Grada Publishing, 2019. ISBN 978-80-271-2208-0.

[6] TROJAN, Stanislav. *Lékařská fyziologie*. Vyd. 4., přeprac. a dopl. Praha: Grada, 2003. ISBN 8024705125.

[7] ROKYTA, Richard. *Fyziologie a patologická fyziologie: pro klinickou praxi*. Praha: Grada Publishing, 2015. ISBN 978-80-247-4867-2.

[8] DORKO, František, Eva VÝBORNÁ a Ján TOKARČÍK. *Základy anatomie pro nelékařské obory: studijní opora*. Ostrava: Ostravská univerzita v Ostravě, 2014. ISBN 978-80-7464-595-2.

[9] TÁBORSKÝ, Miloš, Josef KAUTZNER, Aleš LINHART, Robert HATALA, Eva GONCALVESOVÁ a Peter HLIVÁK, ed. *Kardiologie*. Praha: Česká kardiologická společnost, 2021. ISBN 978-80-271-1439-9.

[10] EISENBERGER, Martin, Alan BULAVA a Martin FIALA. *Základy srdeční elektrofyziologie a katétrových ablací*. Praha: Grada, 2012. ISBN 978-80-247-3677-8.

[11] RANJAN, Rajiv a V. K. GIRI. A Unified Approach of ECG Signal Analysis. *International Journal of Soft Computing and Engineer*ing [online]. Blue Eyes Intelligence Engineering & Sciences Publicatio, 2012, **2**(3), 5-10 [cit. 2022-03-18]. ISSN 2231-2307.

[12] PUURTINEN, Merja, Jari HYTTINEN, Jari VIIK, Pasi KAUPPINEN a Jaakko MALMIVUO. *Modeling bipolar ECG signal strength with thorax models and validation of the modeling method*. 2004. Conference Paper. Tampere University of Technology, Ragnar Granit Institute.

[13] MICHAEL, KEVIN A., BRETT J. PETERSON, ARTHUR M. YUE, et al. Use of an Intracardiac Electrogram Eliminates the Need for a Surface ECG during Implantable Cardioverter-Defibrillator Follow-Up. *Pacing and Clinical Electrophysiology* [online]. 2007, **30**(12), 1432-1437 [cit. 2022-04-17]. ISSN 01478389. Dostupné z: doi:10.1111/j.1540-8159.2007.00888.x

[14] Electrophysiological Testing: Tools and Techniques. *Thoracic Key: Fastest Thoracic Insight Engine* [online]. 2019 [cit. 2022-05-17].

[15] MACEDO, Paula, Sandeep PATEL, Susan BISCO a Samuel ASIRVATHAM. Septal Accessory Pathway: Anatomy, Causes for Difficulty, and an Approach to Ablation. *Indian Pacing and Electrophysiology Journa*l. 2010, **10**(7), 292-309. ISSN 0972-6292.

[16] J. JORDAENS, Luc a Dominic A.M.J. THEUNS. *Implantable Cardioverter Defibrillator Stored ECGs: Clinical Management and Case Reports*. 1. Springer, London. ISBN 978-1-84628-679-7.

[17] Advanced Mapping and Navigation Modalities. *Thoracic Key: Fastest Thoracic Insight Engine* [online]. 2019 [cit. 2022-04-17].

[18] DUKKIPATI, Srinivas, Richard MALLOZZI, Ehud J. SCHMIDT, et al. Electroanatomic Mapping of the Left Ventricle in a Porcine Model of Chronic Myocardial Infarction With Magnetic Resonance–Based Catheter Tracking. *AHA/ASH Journals* [online]. American Heart Association, 4 Aug 2008, 853–862 [cit. 2022-04-17].

[19] ISSA, Ziad F., John M. MILLER a Douglas P. ZIPES. Conventional Intracardiac Mapping Techniques. *Clinical Arrhythmology and Electrophysiology* [online]. Elsevier, 2019, 2019, s. 125-154 [cit. 2022-04-17]. ISBN 9780323523561.

[20] KOZUMPLÍK, Jiří, Radim KOLÁŘ a Jiří JAN. *Číslicové zpracování signálů v prostředí Matlab*. Brno: Vysoké učení technické, 2001. ISBN 80-214-1964-4.

[21] LYONS, G. Richard. *Understanding digital signal processing*. Vyd. 2. New Jersey: Prentice-Hall, 2004. ISBN 0-13-108989-7.

[22] KOZUMPLÍK, J.: *Multitaktní systémy*. Elektronická skripta FEKT VUT, Brno, 2005.

[23] OPPENHEIM, Alan V., Ronald W. SCHAFER a John R. BUCK. *Discrete-time signal processing*. 2nd ed. Upper Saddle River, NJ: Prentice Hall, 1999. ISBN 0-13-754920-2.

[24] ABDULATEEF, Salwa a Mohanad SALMAN. A Comprehensive Review of Image Segmentation Techniques. *Iraqi Journal for Electrical and Electronic Engineering* [online]. 2021, **17**(2), 166-175 [cit. 2022-04-17]. ISSN 2078-6069. Dostupné z: doi:10.37917/ijeee.17.2.18

[25] WALEK, Petr, Martin LAMOŠ a Jiří JAN. *Analýza biomedicínských obrazů*. Analýza biomedicínských obrazů. Brno: 2013. p. 1-138. ISBN: 978-80-214-4792- 9.

[26] SONKA, Milan, Vaclav HLAVAC a Roger BOYLE. Segmentation. SONKA, Milan, Vaclav HLAVAC a Roger BOYLE. *Image Processing, Analysis and Machine Vision* [online]. Boston, MA: Springer US, 1993, 1993, s. 112-191 [cit. 2022-05-17]. ISBN 978-0-412-45570-4.

[27] SANDELIN, Fredrik. *Semantic and Instance Segmentation of Room Features in Floor Plans using Mask R-CNN*. Disciplinary Domain of Science and Technology, Mathematics and Computer Science, Department of Information Technology, 2019. Studentská práce. Uppsala University. Vedoucí práce Klas Sjöberg.

[28] GURUPRASAD, Parthima, Kushal MAHALINGPUR a Manjesh T.N. *Overview of different thresholding methods in image processing*. Bangalore, 2020. Conference Paper. Vivekananda Institute of Technology.

[29] OTSU, Nobuyuki. A Threshold Selection Method from Gray-Level Histograms. *IEEE*. January 1979, **9**(1), 62 - 66. ISSN 0018-9472.

[30] YOUSEFI, *Jamileh. Image Binarization using Otsu Thresholding Algorithm*. Ontario, Canada, 2011. Research. University of Guelph.

[31] KRISHNAN, Muthu. Otsu's method for image thresholding explained and implemented. *Muthukrishnan: AI, Computer Vision and Mathematics* [online]. c2021, March 13, 2020 [cit. 2022-05-17].

[32] B*, Selvapriya a Raghu B. Pseudocoloring Of Medical Images: A Research. *International Journal of Engineering and Advanced Technology* [online]. 2019, **8**(6), 3712-3716 [cit. 2022-04-17]. ISSN 22498958.

[33] SRISHA, Ravi a Am KHAN. *Morphological Operations for Image Processing : Understanding and its Applications: Understanding and its Applications*. 2013. Conference Paper. Mangalore university.

[34] ZEMOURI, Ryad, Noureddine ZERHOUNI a Daniel RACOCEANU. Deep Learning in the Biomedical Applications: Recent and Future Status. *Applied Sciences* [online]. 2019, **9**(8) [cit. 2022-04-17]. ISSN 2076-3417.

[35] KIRANYAZ, Serkan, Turker INCE, Osama ABDELJABER, Onur AVCI a Moncef GABBOUJ. 1-D Convolutional Neural Networks for Signal Processing Applications. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP) [online]. IEEE, 2019, 2019, s. 8360-8364 [cit. 2022-04-17]. ISBN 978-1-4799-8131-1.

[36] BELO, David, João RODRIGUES, João R. VAZ, Pedro PEZARAT-CORREIA a Hugo GAMBOA. Biosignals learning and synthesis using deep neural networks. *BioMedical Engineering OnLine* [online]. 2017, **16**(1) [cit. 2022-05-17]. ISSN 1475-925X.

[37] ZUPAN, Jure. Introduction to Artificial Neural Network (ANN) Methods: What They Are and How to Use Them. *Acta Chimica Slovenica*. Slovenia: Slovenian Chemical Society, [1954-], **41**(3), 327-352. ISSN 1318-0207.

[38] CHMELIK, Jiri. 7. *Artificial Neural Networks: Machine learning*. Department of Biomedical Engineering, 2021.

[39] CHANDRA, Akshay L. Perceptron Learning Algorithm: A Graphical Explanation Of Why It Works. *Towards Data Science* [online]. Aug 22, 2018 [cit. 2022-04-17].

[40] JORDAN, Jeremy. Neural networks: activation functions. *Jeremy Jordan* [online]. c2022, JEREMY JORDAN 4 JUL 2017 [cit. 2022-04-17].

[41] HRISTEV, R. M., 1998. *The ANN Book* [online]. 1. vyd. B.m.: publisher unknown.

[42] FENG, Jianli a Shengnan LU. Performance Analysis of Various Activation Functions in Artificial Neural Networks. *Journal of Physics: Conference Series* [online]. 2019, **1237**(2) [cit. 2022-04-17]. ISSN 1742-6588.

[43] LENDAVE, Vijaysinh. Can ReLU Cause Exploding Gradients if Applied to Solve Vanishing Gradients?. *Analytics India Magazine: Artificial Intelligence, Data Science, Machine Learning* [online]. [cit. 2022-04-17].

[44] SZANDAŁA, Tomasz. Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks. *ArXiv* [online]. [cit. 2022-04-17].

[45] YADAV, Saurabh. Weight Initialization Techniques in Neural Networks. *Towards Data Science* [online]. Nov 9, 2018 [cit. 2022-04-17].

[46] NAWI, Nazri Mohd, Faridah HAMZAH, Norhamreeza Abdul HAMID, Muhammad Zubair REHMAN, Mohammad AAMIR a Azizul Ramli AZHAR. An Optimized Back Propagation Learning Algorithm with Adaptive Learning Rate. *International Journal on Advanced Science, Engineering and Information Technology* [online]. 2017, **7**(5), 1693-1700 [cit. 2022-04-17]. ISSN 2460-6952.

[47] SHARMA, Sagar. Epoch vs Batch Size vs Iterations. *Towards Data Science* [online]. Sep 23, 2017 [cit. 2022-04-18].

[48] ROY, Abhijit. An Introduction to Gradient Descent and Backpropagation. *Towards Data Science* [online]. [cit. 2022-04-18].

[49] DU, Simon S., Jason D. LEE, Haochuan LI, Liwei WANG a Xiyu ZHAI. Gradient Descent Finds Global Minima of Deep Neural Networks. *Proceedings of Machine Learning Research* [online]. [cit. 2022-04-18]. ISSN 2640-3498.

[50] YING, Xue. An Overview of Overfitting and its Solutions. *Journal of Physics: Conference Series* [online]. 2019, **1168** [cit. 2022-04-17]. ISSN 1742-6588.

[51] GHOJOGH, Benyamin a Mark CROWLEY. The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial. *ArXiv* [online]. 28 May 2019 [cit. 2022-04-18].

[52] PARMAR, Ravindra. Common Loss functions in machine learning. *Towards Data Science* [online]. Sep 2, 2018 [cit. 2022-04-18].

[53] GODOY, Daniel. Understanding binary cross-entropy / log loss: a visual explanation. *Towards Data Science* [online]. Nov 21, 2018 [cit. 2022-04-18].

[54] ZHOU, Ziyun, Hong HUANG a Binhao FANG. Application of Weighted Cross-Entropy Loss Function in Intrusion Detection. *Journal of Computer and Communication*s [online]. 2021, **09**(11), 1-21 [cit. 2022-04-18]. ISSN 2327-5219.

[55] RUDER, Sebastian. An overview of gradient descent optimization algorithms. *Sebastian Ruder* [online]. c2022, 19 JAN 2016 [cit. 2022-04-18].

[56] HAVRLANT, Lukáš. Průběh funkce: extrémy. *Učebnice matematiky — Matematika polopatě* [online]. 2006—2022 [cit. 2022-04-18].

[57] KRUPKOVÁ, Vlasta a Petr FUCHS. *Matematika 1: (Elektrotechnika, elektronika, komunikační a řídicí technika)* [online]. Ústav matematiky FEKT VUT v Brně, 2014 [cit. 2022-04-18].

[58] WEISSTEIN, Eric W. Saddle Point: from Wolfram MathWorld. *Wolfram MathWorld: The Web's Most Extensive Mathematics Resource* [online]. Wolfram Research, c1999–2022 [cit. 2022-04-18].

[59] POPESCU, Marius-Constantin, Liliana PERESCU-POPESCU, Valentina Emilia BALAS a Nikos E MASTORAKIS. Multilayer Perceptron and Neural Networks. *WSEAS TRANSACTIONS on CIRCUITS and SYSTEMS*. 2009, **8**(7), 579-588. ISSN 1109-2734.

[60] HASSAN, Hassan Mohamed, Abdelazim M NEGM, Mohamed ZAHRAN a Oliver SAAVEDRA. Assessment Of Artificial Neural Network For Bathymetry Estimation Using High Resolution Satellite Imagery In Shallow Lakes: Case Study El Burullus Lake. *International Water Technology Journal*, IWTJ. 2015, **5**(4), 248-259. ISSN 2090-5440.

[61] PASUPA, Kitsuchart a Wisuwat SUNHEM. A comparison between shallow and deep architecture classifiers on small dataset. In: *2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE)* [online]. IEEE, 2016, 2016, s. 1-6 [cit. 2022-04-18]. ISBN 978-1-5090-4139-8.

[62] SAHU, Madhusmita a Rasmita DASH. A Survey on Deep Learning: Convolution Neural Network (CNN). MISHRA, Debahuti, Rajkumar BUYYA, Prasant MOHAPATRA a Srikanta PATNAIK, ed. *Intelligent and Cloud Computing* [online]. Singapore: Springer Singapore, 2021, 2021-08-29, s. 317-325 [cit. 2022-04-18]. Smart Innovation, Systems and Technologies. ISBN 978-981-15-6201-3.

[63] CHMELIK, Jiri. 8. *Deep Learning 1: Machine learning*. Department of Biomedical Engineering, 2021.

[64] DAVID, Omid E. a Nathan S. NETANYAHU. DeepPainter: Painter Classification Using Deep Convolutional Autoencoders. VILLA, Alessandro E.P., Paolo MASULLI a Antonio Javier PONS RIVERO, ed. *Artificial Neural Networks and Machine Learning – ICANN 2016* [online]. Cham: Springer International Publishing, 2016, 2016-08-13, s. 20-28 [cit. 2022-04-18]. Lecture Notes in Computer Science. ISBN 978-3-319-44780-3.

[65] BEZDAN, Timea a Nebojša BAČANIN DŽAKULA. Convolutional Neural Network Layers and Architectures. In: *Proceedings of the International Scientific Conference - Sinteza 2019* [online]. Novi Sad, Serbia: Singidunum University, 2019, 2019, s. 445-451 [cit. 2022-04-18]. ISBN 978-86-7912-703-7.

[66] HOANG, Hong Hai a Hoang Hieu TRINH. Improvement for Convolutional Neural Networks in Image Classification Using Long Skip Connection. *Applied Sciences* [online]. 2021, **11**(5) [cit. 2022-05-19]. ISSN 2076-3417.

[67] WU, Bichen, Alvin WAN, Xiangyu YUE, et al. Shift: A Zero FLOP, Zero Parameter Alternative to Spatial Convolutions. In*: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* [online]. IEEE, 2018, 2018, s. 9127-9135 [cit. 2022-04-18]. ISBN 978-1-5386-6420-9.

[68] KUNDU, Souvik, Hesham MOSTAFA, Sharath Nittur SRIDHAR a Sairam SUNDARESAN. Attention-based Image Upsampling. *ArXiv* [online]. 1-10 [cit. 2022-05-19].

[69] T.DANG, Anh. Top 10 CNN Architectures Every Machine Learning Engineer Should Know: From 2012 to present. *Towards Data Science* [online]. Jan 5, 2021 [cit. 2022-04-18].

[70] KIRANYAZ, Serkan, Onur AVCI, Osama ABDELJABER, Turker INCE, Moncef GABBOUJ a Daniel J. INMAN. 1D convolutional neural networks and applications: A survey. Mechanical Systems and Signal Processing [online]. 2021, **151** [cit. 2022-04-18]. ISSN 08883270.

[71] RONNEBERGER, Olaf, Philipp FISCHER a Thomas BROX. U-Net: Convolutional Networks for Biomedical Image Segmentation. NAVAB, Nassir, Joachim HORNEGGER, William M. WELLS a Alejandro F. FRANGI, ed. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* [online]. Cham: Springer International Publishing, 2015, 2015-11-18, s. 234-241 [cit. 2022-04-18]. Lecture Notes in Computer Science. ISBN 978-3-319-24573-7.

[72] AUDEBERT, Nicolas, Bertrand LE SAUX a Sébastien LEFÈVRE. Segment-before-Detect: Vehicle Detection and Classification through Semantic Segmentation of Aerial Images. *Remote Sensing* [online]. 2017, **9**(4) [cit. 2022-05-18]. ISSN 2072-4292.

[73] HUSSEIN, Sarah a Khawla ALI. Semantic Segmentation of Aerial Images Using U-Net Architecture. *Iraqi Journal for Electrical and Electronic Engineering* [online]. 2021, **18**(1), 58-63 [cit. 2022-04-18]. ISSN 2078-6069.

[74] COSTANDY, Rana N., Safa M. GASSER, Mohamed S. EL-MAHALLAWY, Mohamed W. FAKHR a Samir Y. MARZOUK. P-Wave Detection Using a Fully Convolutional Neural Network in Electrocardiogram Images. *Applied Sciences* [online]. 2020, **10**(3) [cit. 2022-04-18]. ISSN 2076-3417.

[75] NULL, Linda a Julia LOBUR. *The essentials of computer organization and architecture*. 2nd ed. Sudbury: Jones and Bartlett Publishers, 2006. ISBN 9780763737696.

[76] PARRILL, Leo. *What is DPI, and why is it important?*. Newegg Insider [online]. Newegg, c2000-2022 [cit. 2022-04-18].

[77] *TensorFlow* [online]. [cit. 2022-04-18].

[78] *Keras: Simple. Flexible. Powerful.* [online]. [cit. 2022-04-18].

[79] BOULILA, Wadii, Maha DRISS, Mohamed AL-SAREM, Faisal SAEED a Moez KRICHEN. Weight Initialization Techniques for Deep Learning Algorithms in Remote Sensing: Recent Trends and Future Perspectives. *ArXiv*.

[80] SHI, Jiyuan, Ji DANG, Mida CUI, Rongzhi ZUO, Kazuhiro SHIMIZU, Akira TSUNODA a Yasuhiro SUZUKI. Improvement of Damage Segmentation Based on Pixel-Level Data Balance Using VGG-Unet. *Applied Sciences* [online]. 2021, **11**(2) [cit. 2022-04-18]. ISSN 2076-3417.

[81] MISHRA, Saloni. Baffling Concept of True Positive and True Negative: Understanding TP/TN and FP/FN. *Towards Data Science* [online]. Jun 4, 2021 [cit. 2022-04-18].

[82] JORDAN, Jeremy. Evaluating image segmentation models. *Jeremy Jordan* [online]. 30 MAY 2018 [cit. 2022-04-18].

[83] TAS, Seyma. How To Evaluate Image Segmentation Models?: Dice and Jaccard's explained…. *Towards Data Science* [online]. [cit. 2022-04-18].

[84] NGUYEN, M. P., C. SCHILLING a O. DÖSSEL. A new approach for automated location of active segments in intracardiac electrograms. DÖSSEL, Olaf a Wolfgang C. SCHLEGEL, ed. World Congress on Medical Physics and Biomedical Engineering, September 7 - 12, 2009, Munich, Germany [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, 2009, s. 763-766 [cit. 2022-04-18]. IFMBE Proceedings. ISBN 978-3-642-03881-5.

[85] PAN, Jiapu a Willis J. TOMPKINS. A Real-Time QRS Detection Algorithm. *IEEE Transactions on Biomedical Engineering* [online]. 1985, **BME-32**(3), 230-236 [cit. 2022-04-18]. ISSN 0018-9294.

SEZNAM ZKRATEK

Adam	Adaptive moment estimation
AP	Anterior posterior
AV	Atrioventrikulární
CNN	Konvoluční neuronová síť
CS	Koronární sinus
DPI	Dots per inch
EKG	Elektrokadiografie
FN Brno	Fakultní nemocnice Brno
FN	False negative
FP	False positive
His	Hisův svazek
iEKG	Intrakardiální EKG
JSON	JavaScript Object Notation
LAO	Left anterior oblique
MAE	Mean absolute error
MRI	Magnetická rezonance
MSE	Mean square error
NLEO	Teagerův nelineární energetický operátor
RA	Right atrium
RAM	Random access memory
RAO	Right anterior oblique
ReLU	Rectified linear unit
RGB	Red green blue
SA	Sinoatriální
TN	True negative
TP	True positive
XML	Extensible Markup Language

SEZNAM PŘÍLOH

Příloha A - Výstupy praktické části

A.1 Tabulka zhodnocení segmentace | U-net

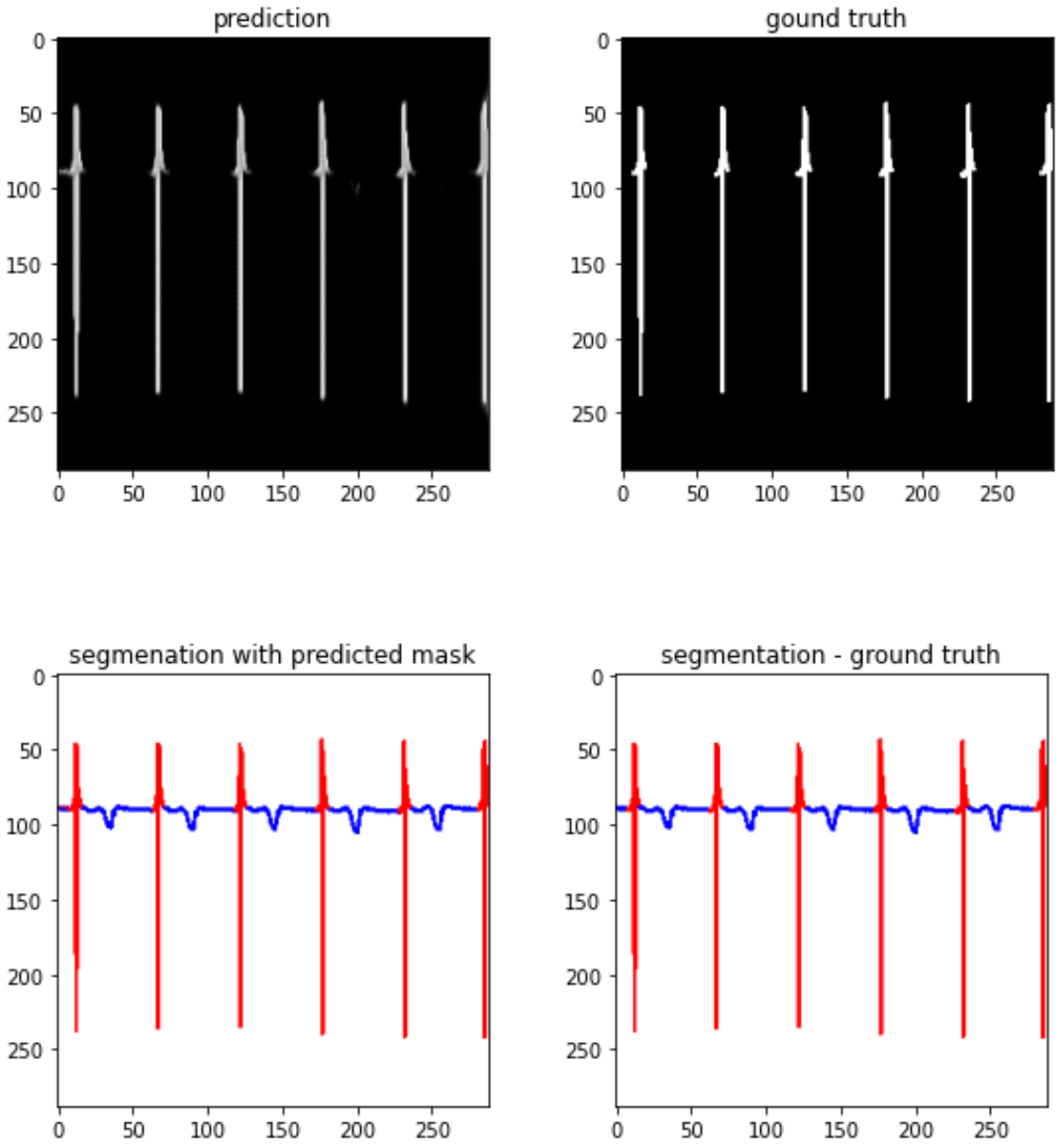
Obraz č.	TP	TN	FP	FN	Preciznost	Sensitivita	Dice
1	4034	78830	46	31	0.989	0.992	0.990
2	3832	77323	1761	28	0.685	0.992	0.811
3	3574	78846	439	85	0.891	0.977	0.932
4	3569	79223	56	96	0.985	0.974	0.979
5	4296	78266	294	88	0.940	0.980	0.957
6	4365	77955	534	90	0.891	0.980	0.933
7	4005	77642	1207	90	0.768	0.978	0.861

A.2 Tabulka zhodnocení segmentace | SegNet

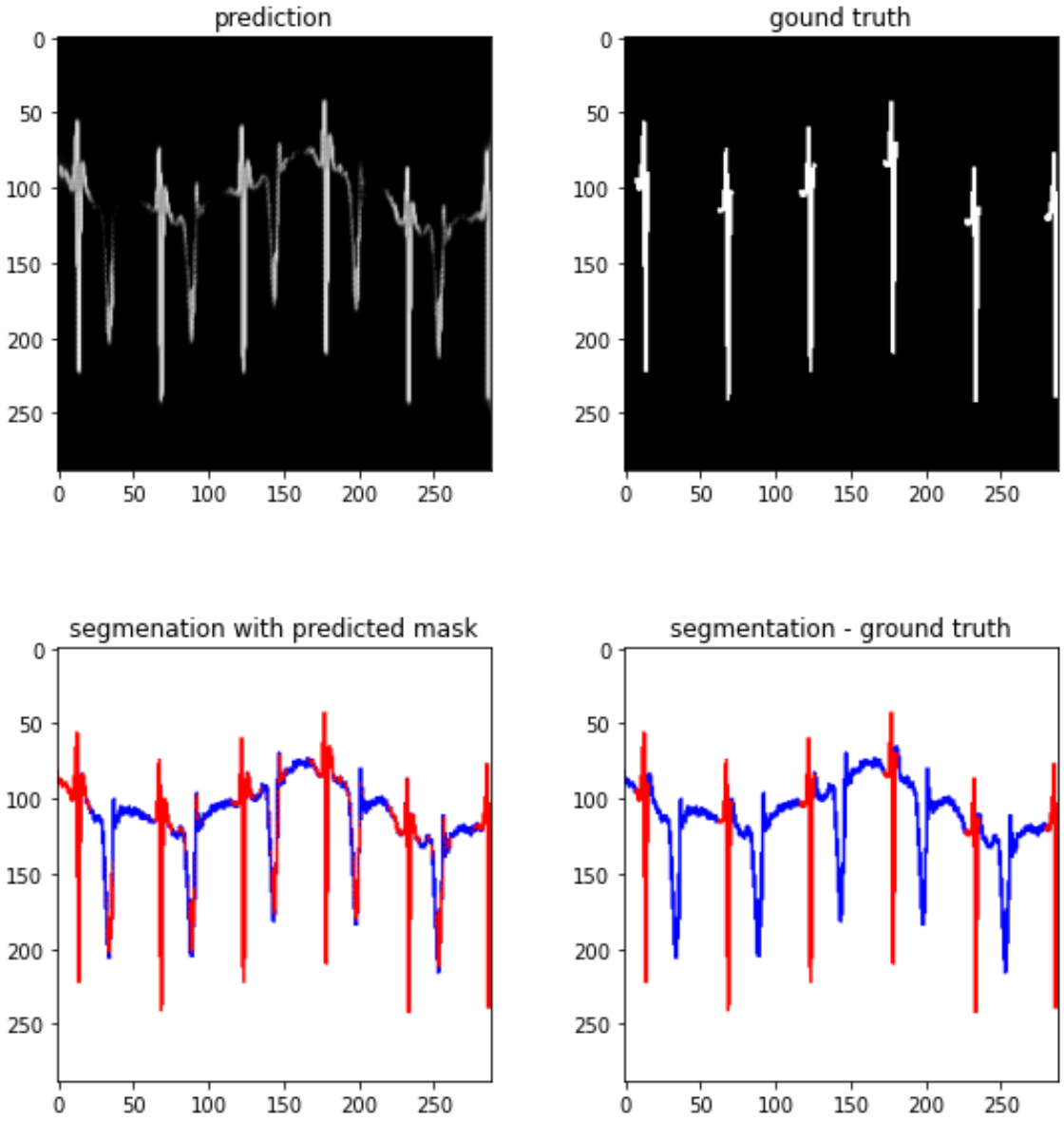
Obraz č.	TP	TN	FP	FN	Preciznost	Sensitivita	Dice
1	1403	70242	8634	2665	0.140	0.345	0.199
2	1661	66849	12235	2199	0.120	0.430	0.187
3	2014	50790	28495	1645	0.066	0.550	0.118
4	1667	73484	5795	1998	0.223	0.455	0.300
5	1915	72840	5720	2469	0.251	0.437	0.320
6	2293	53796	24693	2162	0.085	0.515	0.146
7	1880	67974	10875	2215	0.147	0.459	0.223

A.3 Výstupy segmentace provedené na testovací množině dat | U-net

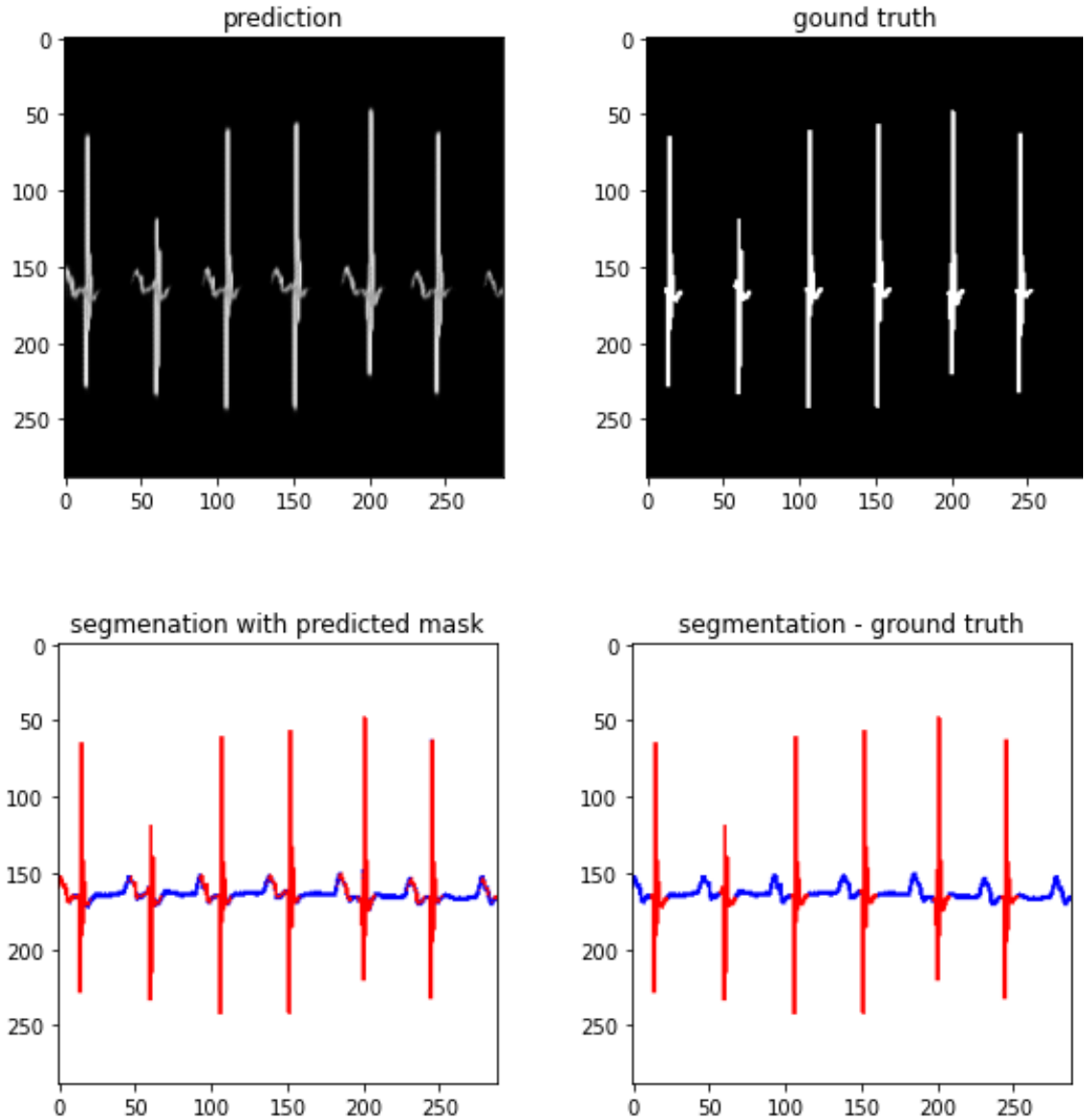
Testovací obraz č.1



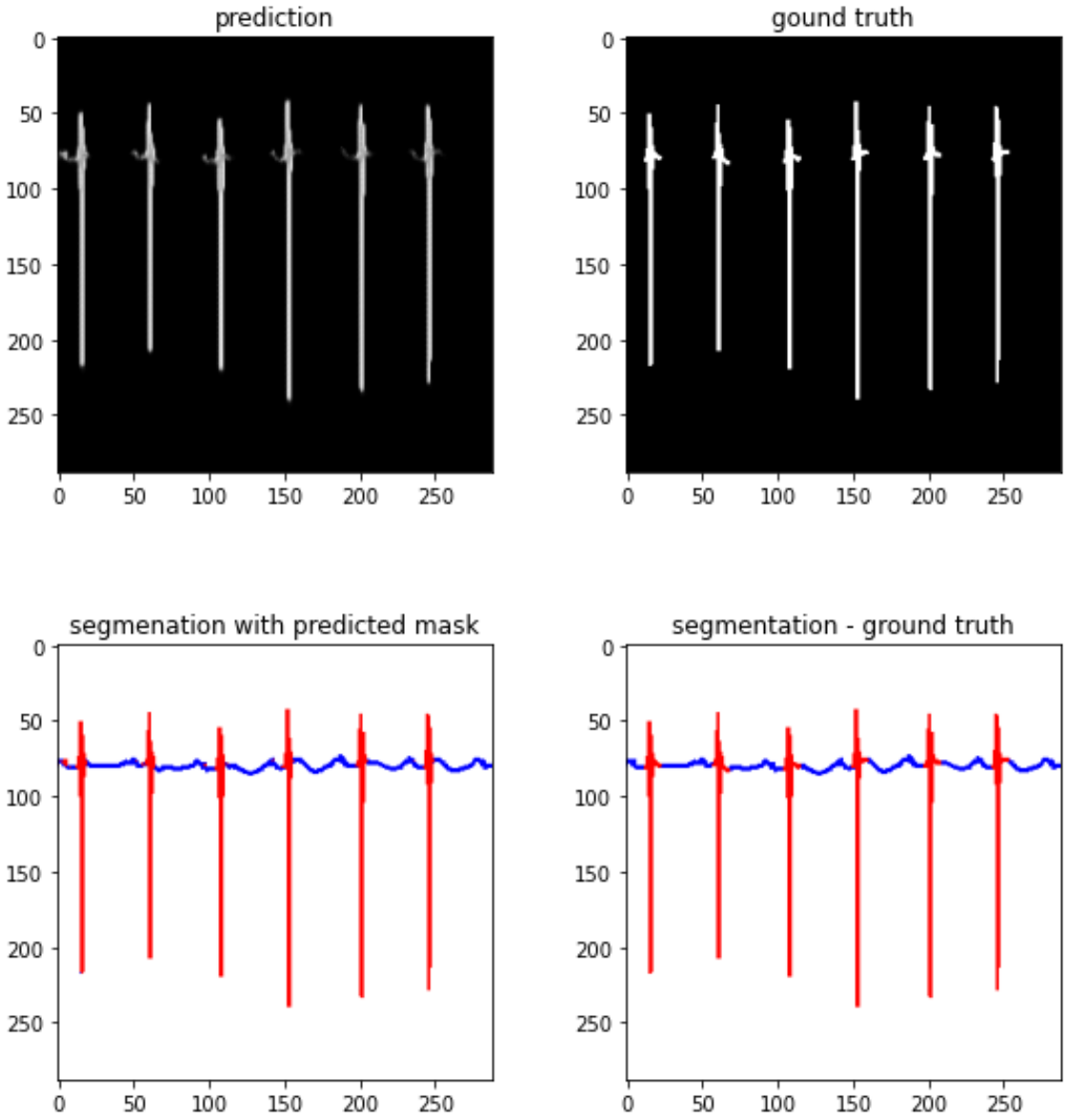
Testovací obraz č.2



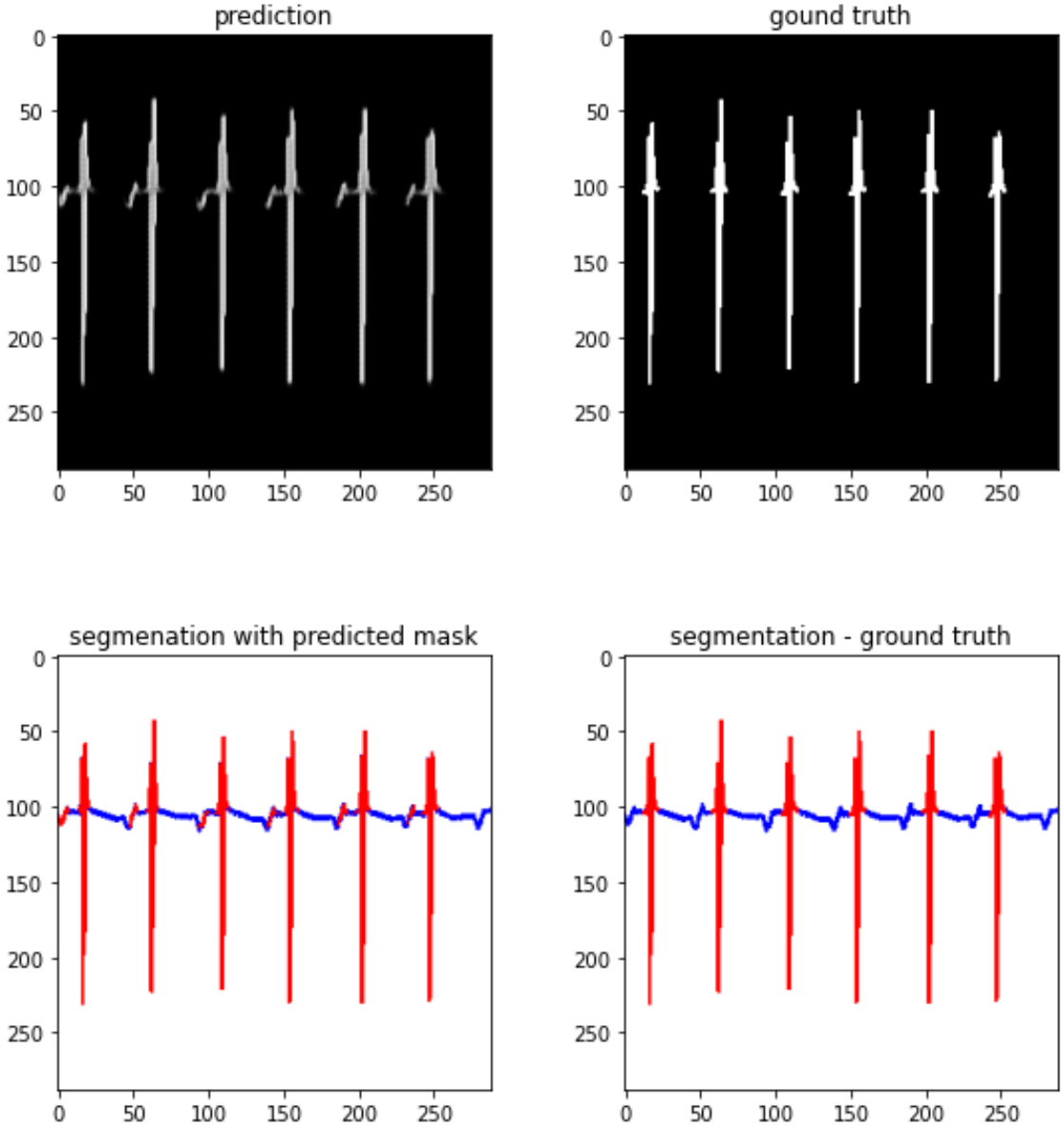
Testovací obraz č.3



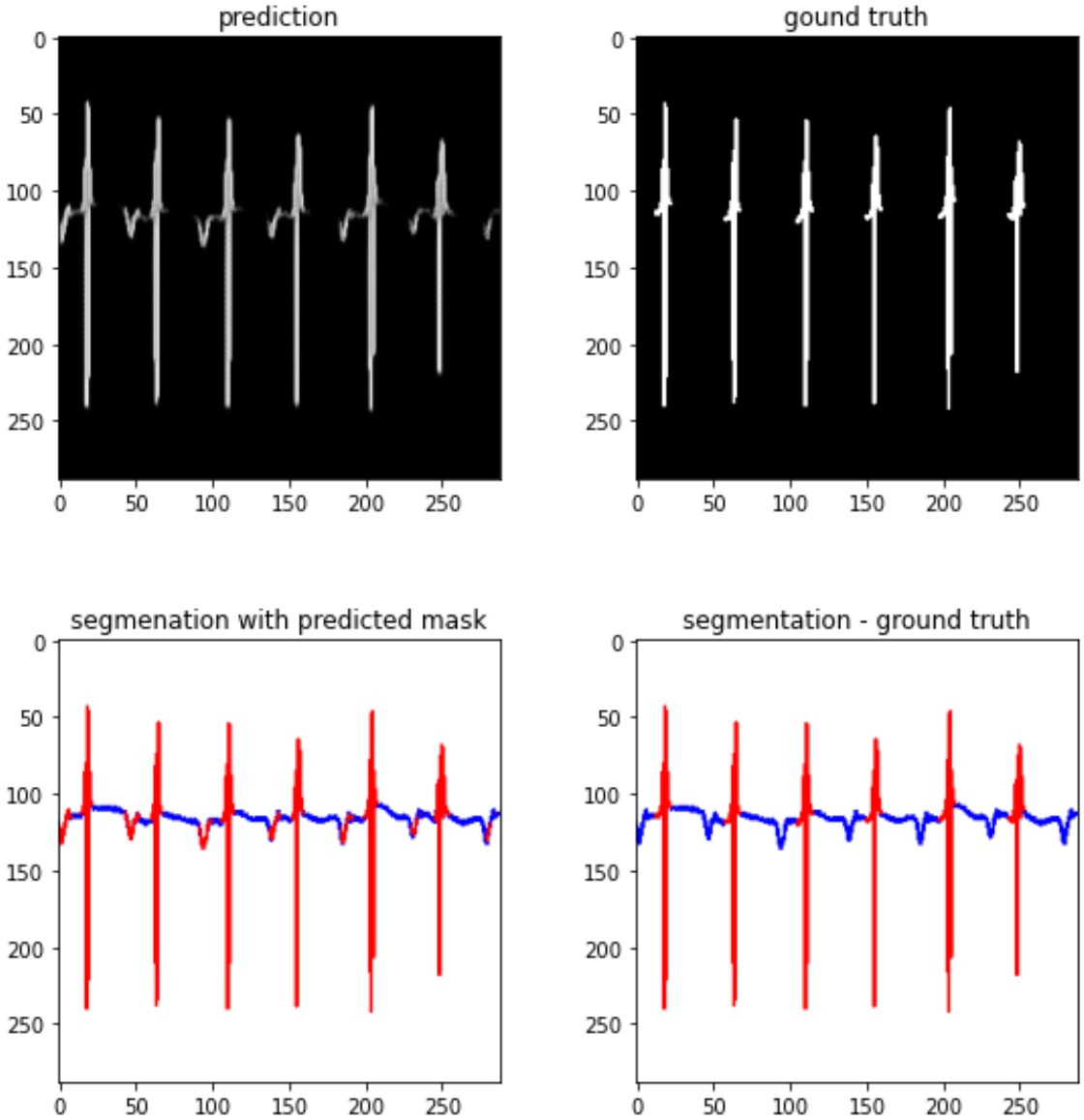
Testovací obraz č.4



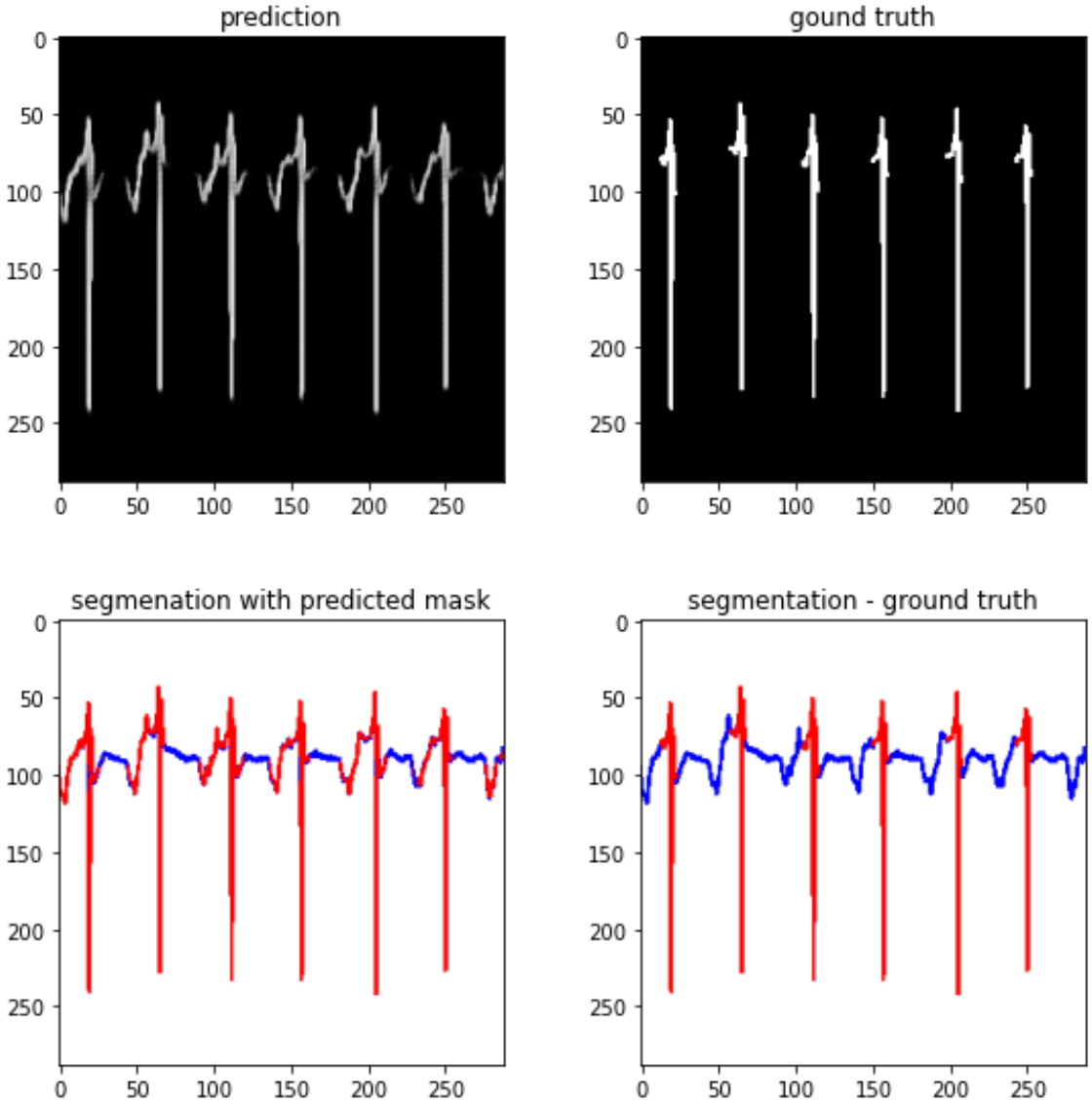
Testovací obraz č.5



Testovací obraz č.6

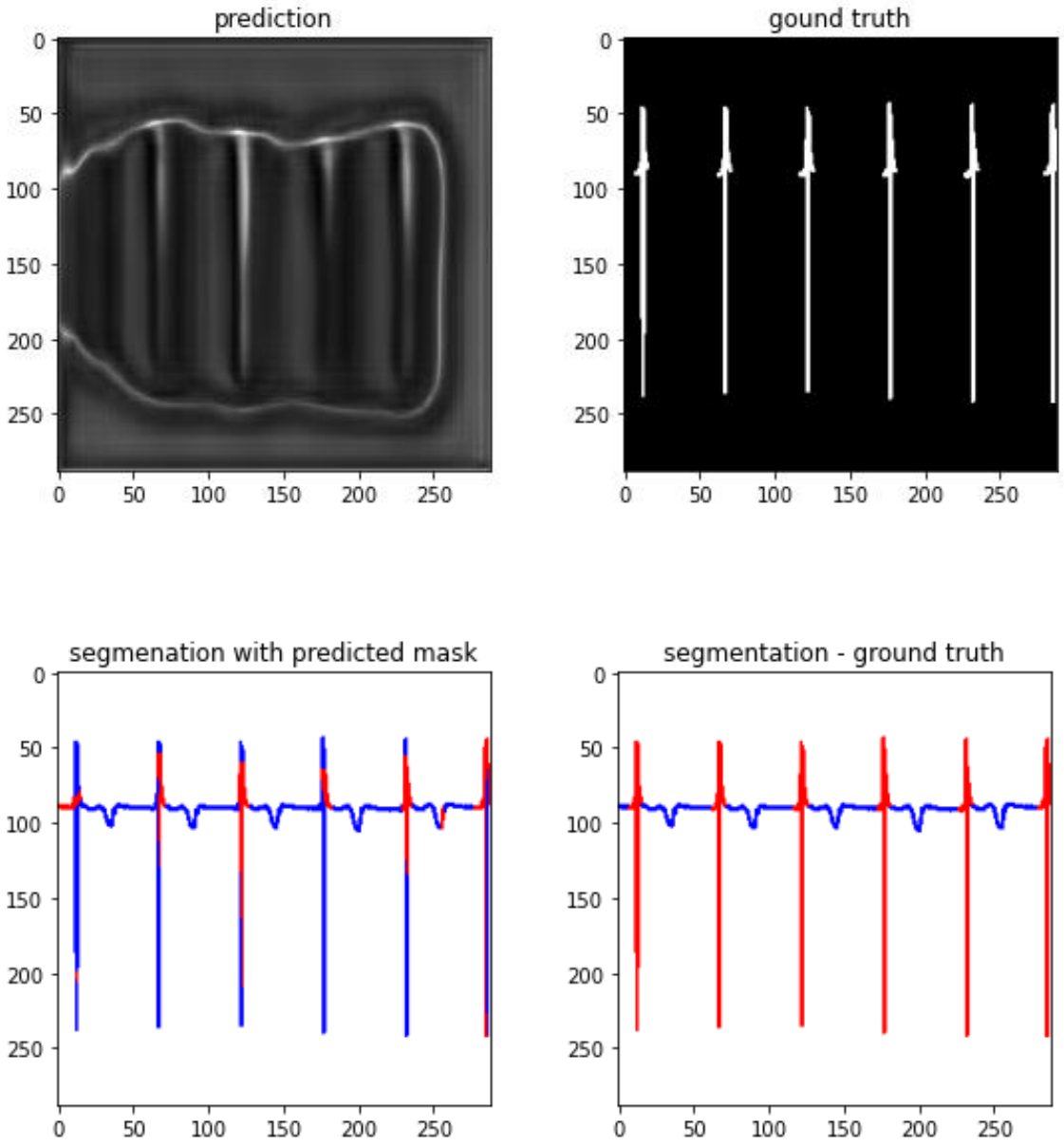


Testovací obraz č.7

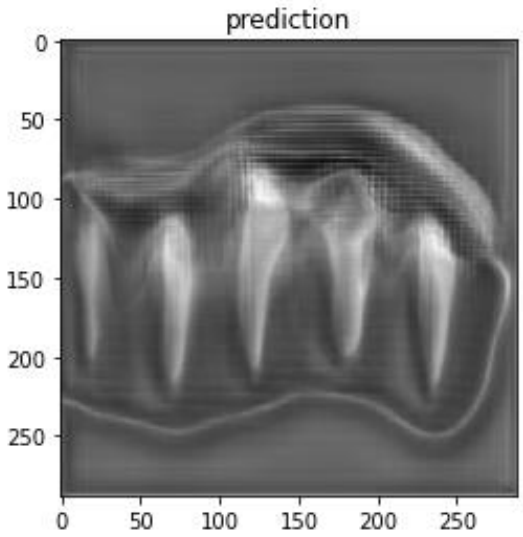


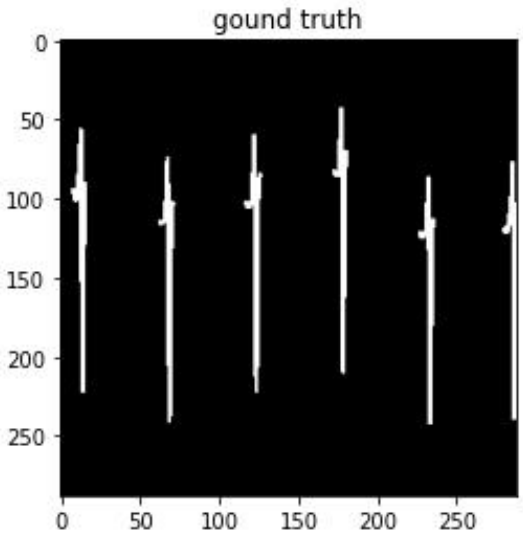
A.4 Výstupy segmentace provedené na testovací množině dat | SegNet

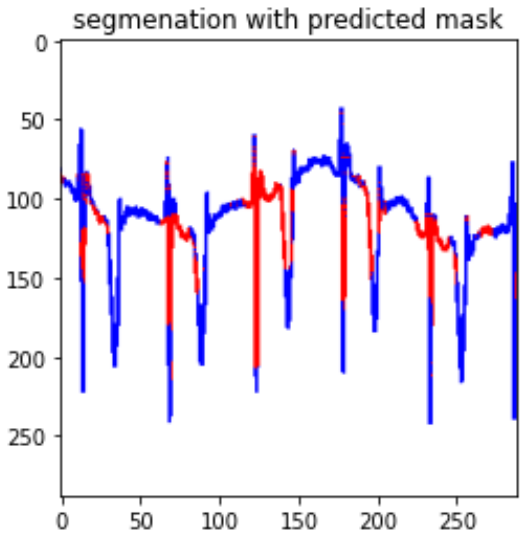
Testovací obraz č.1

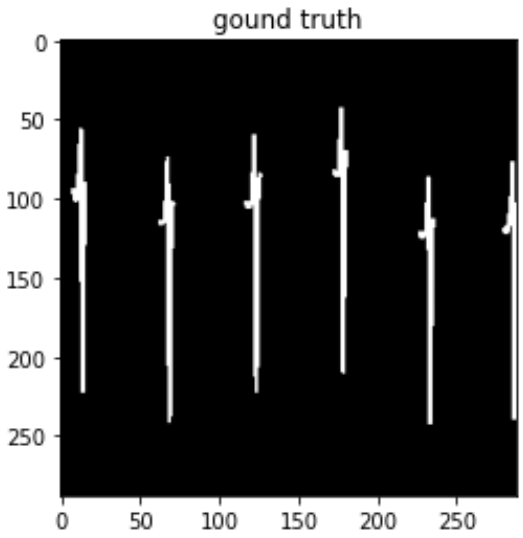


Testovací obraz č.2

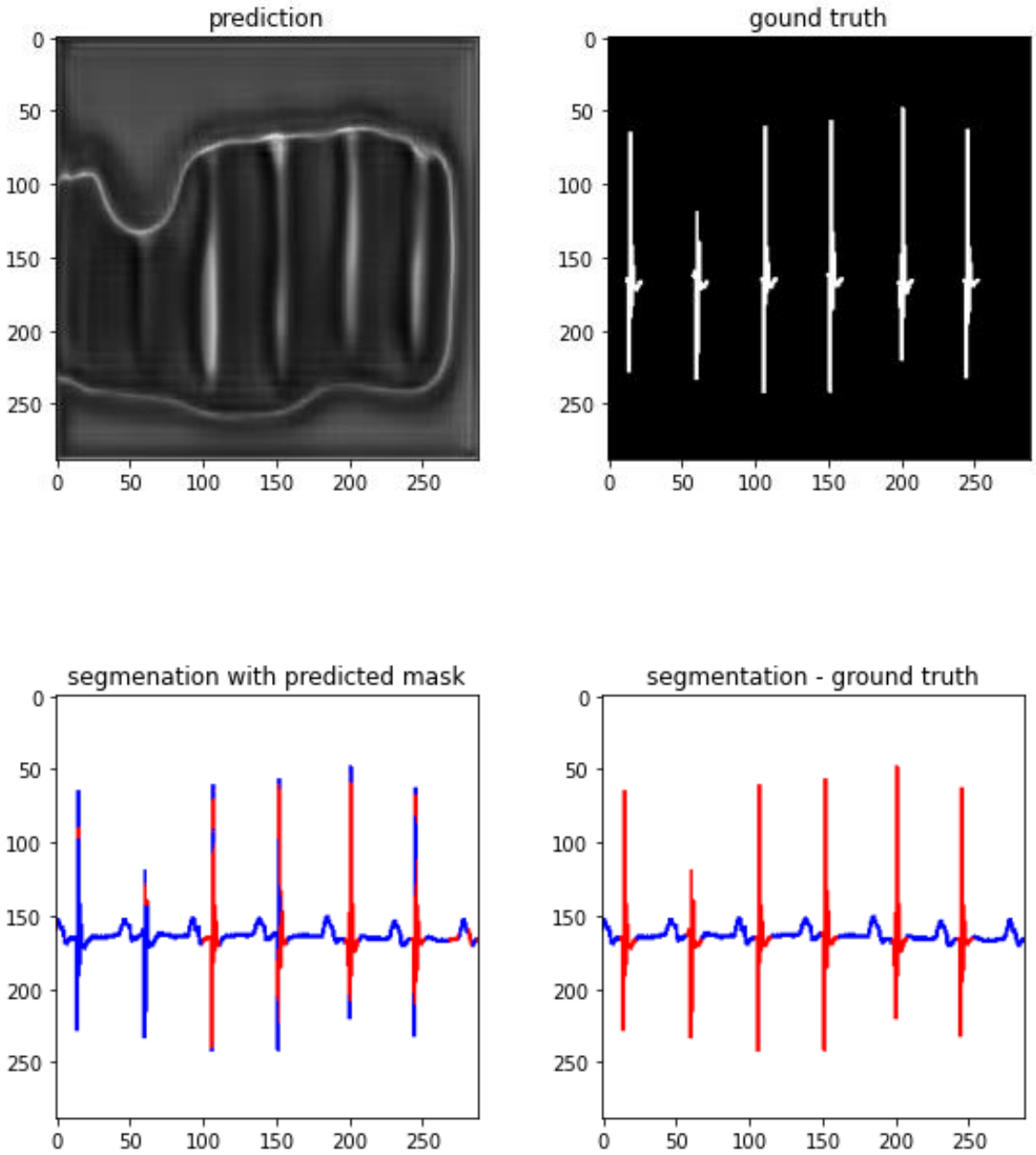




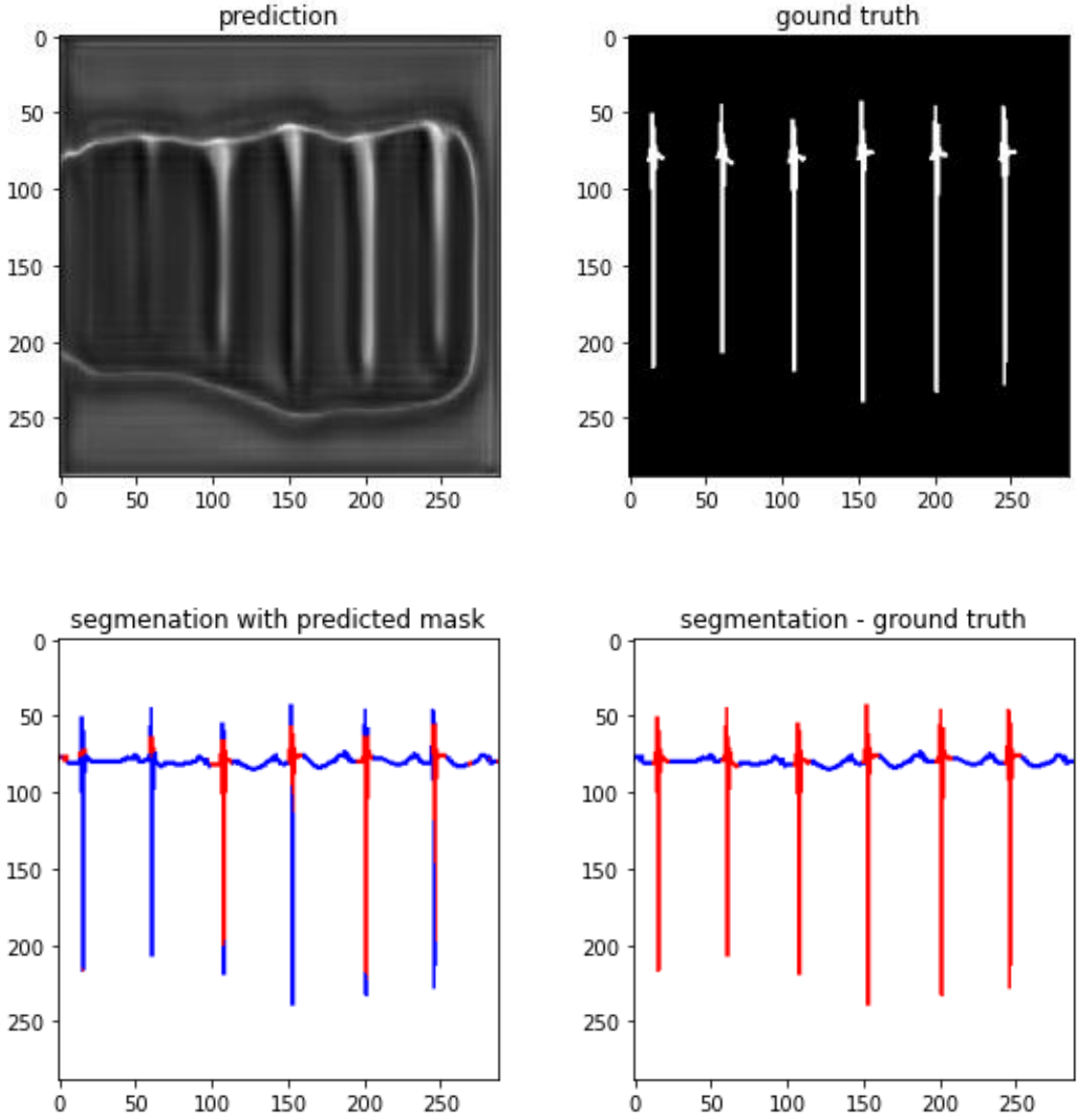




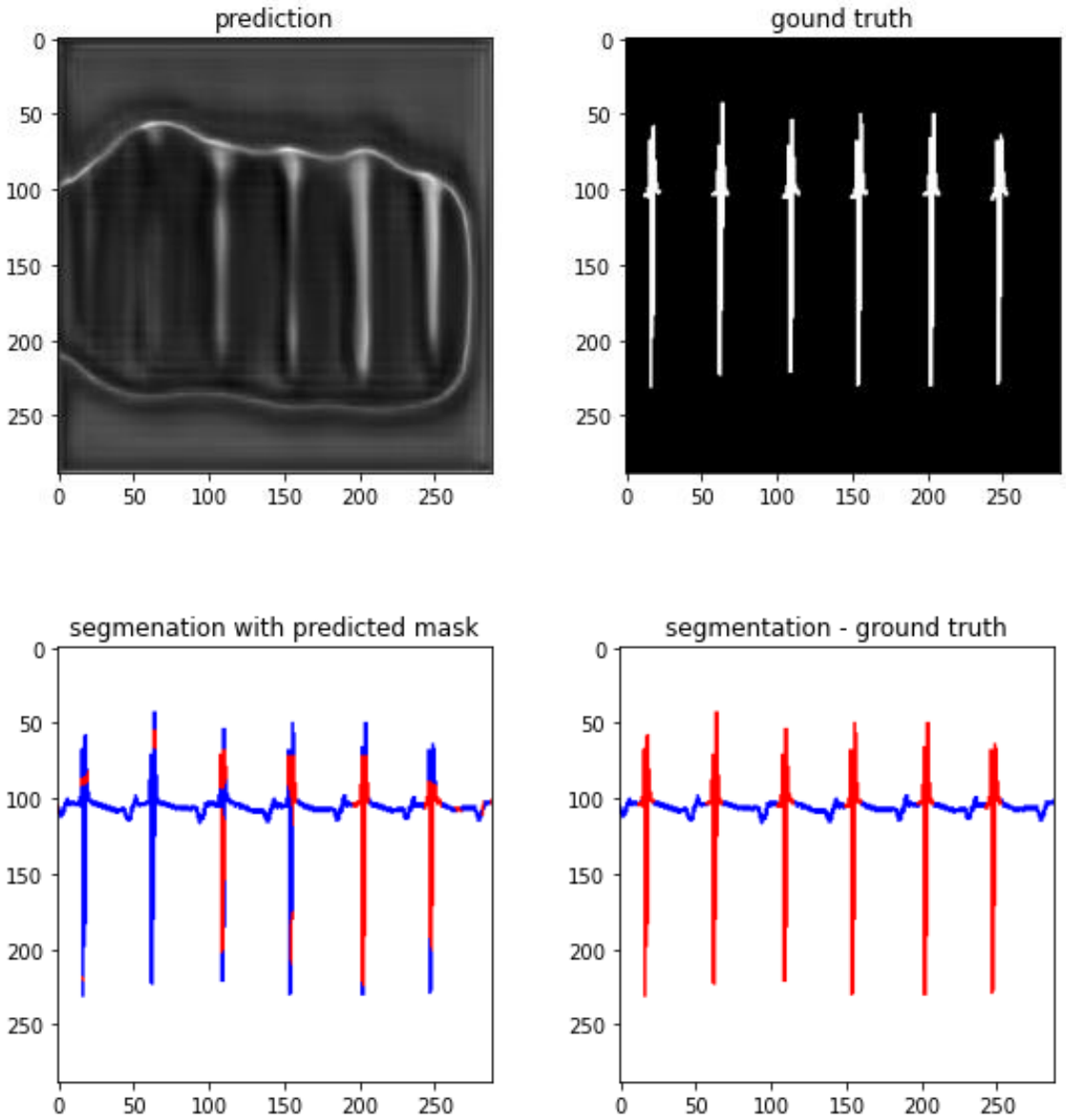
Testovací obraz č.3



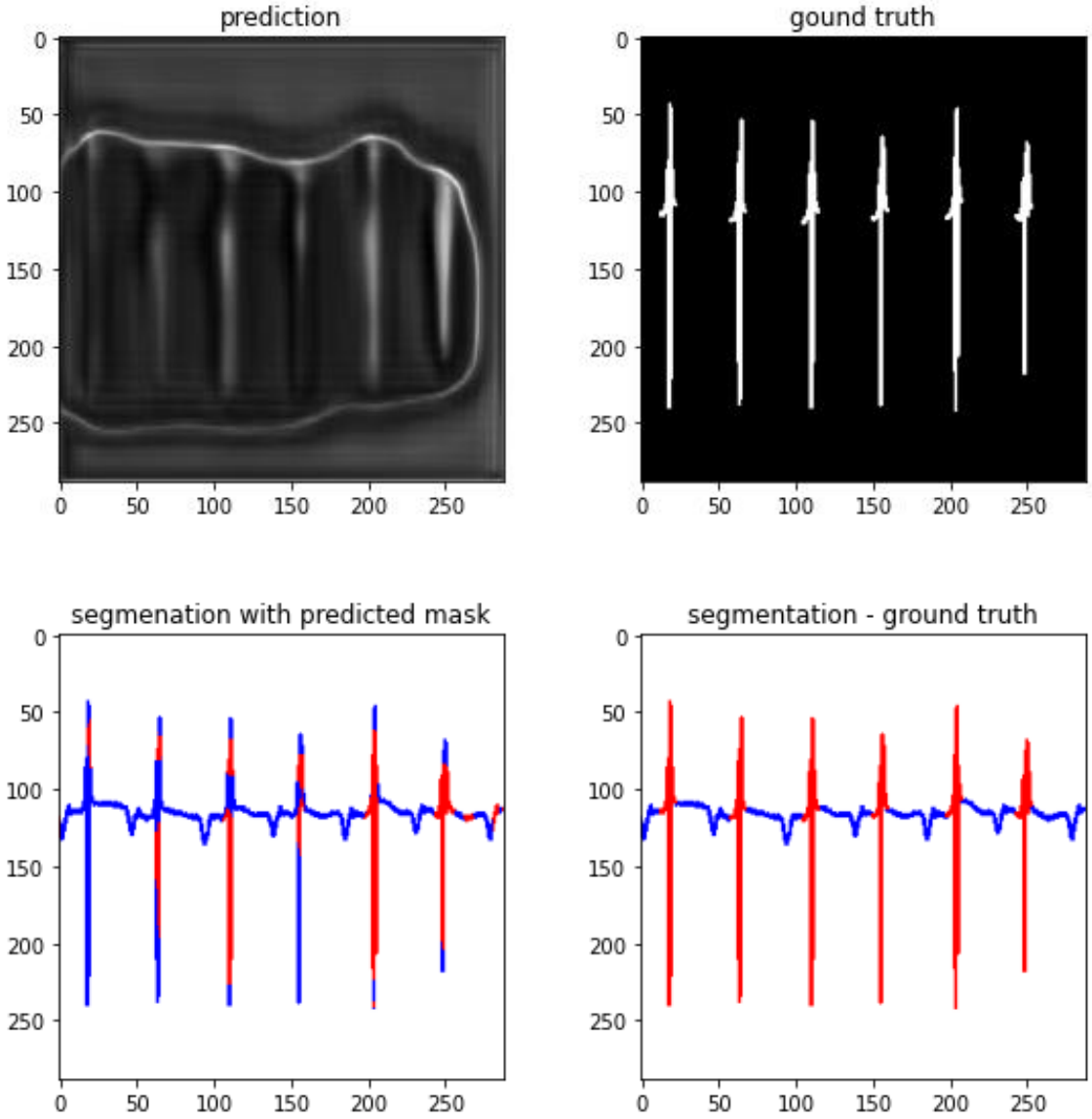
Testovací obraz č.4



Testovací obraz č.5



Testovací obraz č.6



Testovací obraz č.7