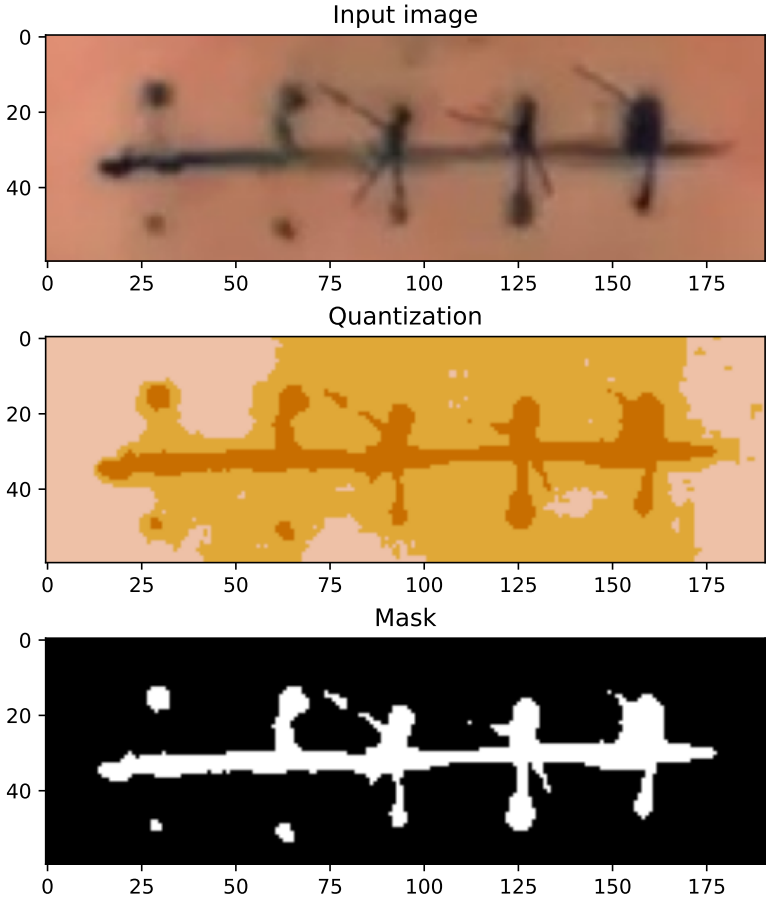
1 Řešení

Nejdříve byl vytvořen Pytorch dataset, jehož metoda *__get_item__(image id)* poskytuje pro dané id obrázku samotný obrázek, jeho masku a počet stehů. Dataset byl následně rozdělen na trénovací a validační část v poměru 90/10. Rozdělení dat se provádí náhodně a je klíčové pro vyhodnocení, neboť některé obrázky datasetu nejsou příliš kvalitní a vyhodnocení na takových obrázcích pak snižuje výsledné přesnosti použitých metod detekce. Rozdělení dat bylo možné ovlivňovat parametrem *seed*, který byl v rámci vyhodnocení výsledků v této práci nastaven na hodnotu 100.

1.1 Segmentace stehu

Toto řešení netrpí problémy způsobenými nekonstantním osvícením scény, které zabraňují užití běžného prahování, a obecně různorodým zbarvením jednotlivých prvků v obrázku. Obrázek kvantizujeme na 3 barvy, což nám scénu rozdělí na tři masky - řez + stehy, první a druhou část pozadí. V dalším kroku se vybere maska stehu na základě detekce horizontálních přímek - pozadí jich teoreticky obsahuje velké množství, zatímco steh obsahuje pouze malé množství, nejčastěji jednu. Tímto získáme masku stehu, na kterou lze aplikovat další metody.



Obrázek 1: Ukázka získaní masky.

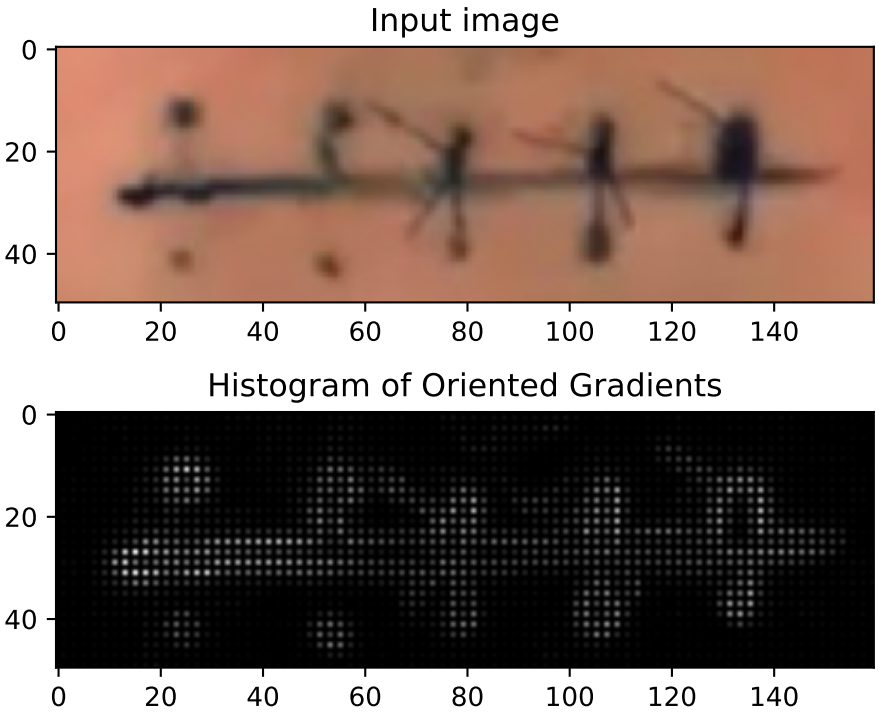
1.2 Konvoluční neuronová síť

Pomocí knihovny *Pytorch* byla navržena konvoluční síť s třemi konvolučními vrstvami s následným poolingem, třemi dopřednými fully-connected vrstvami s osmi výstupními neurony. Celkem má taková síť přibližně milion parametrů. Rozměry vstupních obrázků byly změněny na (50, 180) a na data byla v průběhu trénování aplikována řada augmentací jako například horizontální a vertikální flip, pootočení nebo změna kontrastu.

Při trénování bylo vyzkoušena řada parametrů jako například různá konstanta učení, *batch_size* nebo *dropout* pravděpodobnost. Konstanta učení byla nakonec nastavena na hodnotu 0.0002 a *dropout* na hodnotu 0.2, neboli 20%. Síť byla trénována pro 2000 epoch s *batch_size* = 128 a na validační části datasetu dosahuje přesnosti 78%. Bylo také vyzkoušeno trénovat síť na maskách získaných v kapitole 1.1. Řešení ale nedosahovalo lepších výsledků než použití sítě na samotné obrázky.

1.3 HOG a klasifikátor

V rámci druhého řešení byl použit HOG (Histogram of Oriented Gradients) deskriptor, který pro každý obrázek získal reprezentativní vektor. Vektory se stejnými počty stehů by si pak měly být podobné a lze podle nich klasifikovat. Pro získané vektory bylo natrénováno několik klasifikátorů z knihovny *sklearn*, přičemž nejlepších výsledků dosahoval klasifikátor SVM. Na validační množině dosáhnul přesnosti 42%. I pro tento způsob bylo vyzkoušeno využít masky z kapitoly 1.1, ale také nebylo dosaženo lepších výsledků.

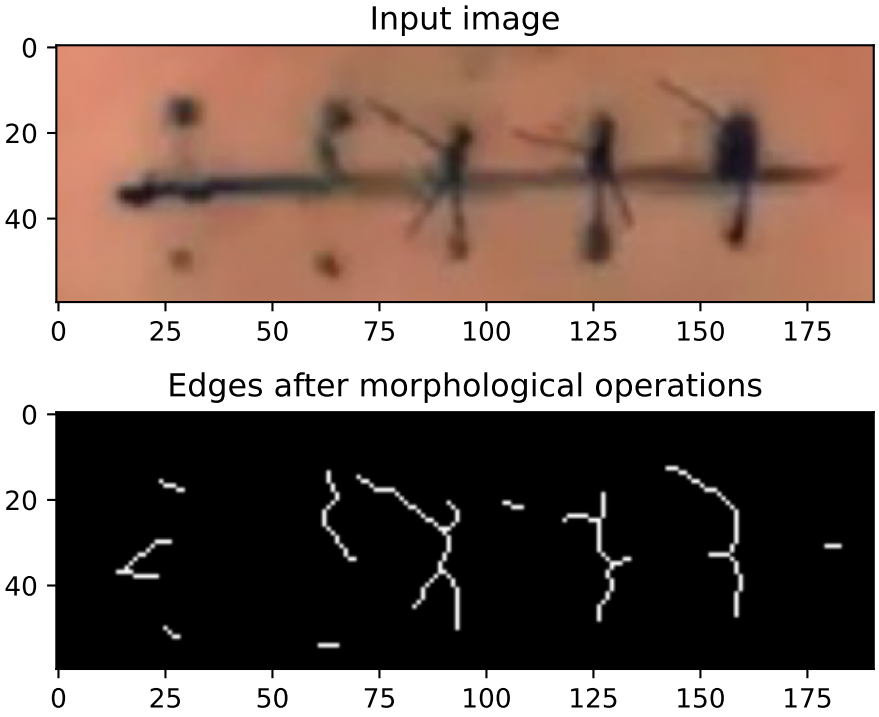


Obrázek 2: Vizualizace funkcionality HOG deskriptoru.

1.4 Hranový detektor

Tento přístup řešení spočíval v myšlence, že pokud bychom z obrázku odstranili vodorovné hrany, pak v místech, kde se nachází stehy, budou delší hrany než v místech, kde stehy nejsou.

Na vstupní jsme nejdříve aplikovali Canny detektor hran. Následovaly morfologické operace uzavření, vyplnění děr a ztenčení čar. Pomocí morfologické operace otevření jsme dále našli dlouhé vodorovné hrany a odečetli je od hran nalezených Canny detektorem. Dilatací jsme dalé spojili rozpojené vertikální hrany a za stehy prohlásili všechny hrany v obrázku delší než stanovený práh pomocí metody *findContours()*. Tento přístup dosahoval na validační části datasetu přesnosti 42%.



Obrázek 3: Nalezené hrany po aplikaci morfologických operací.

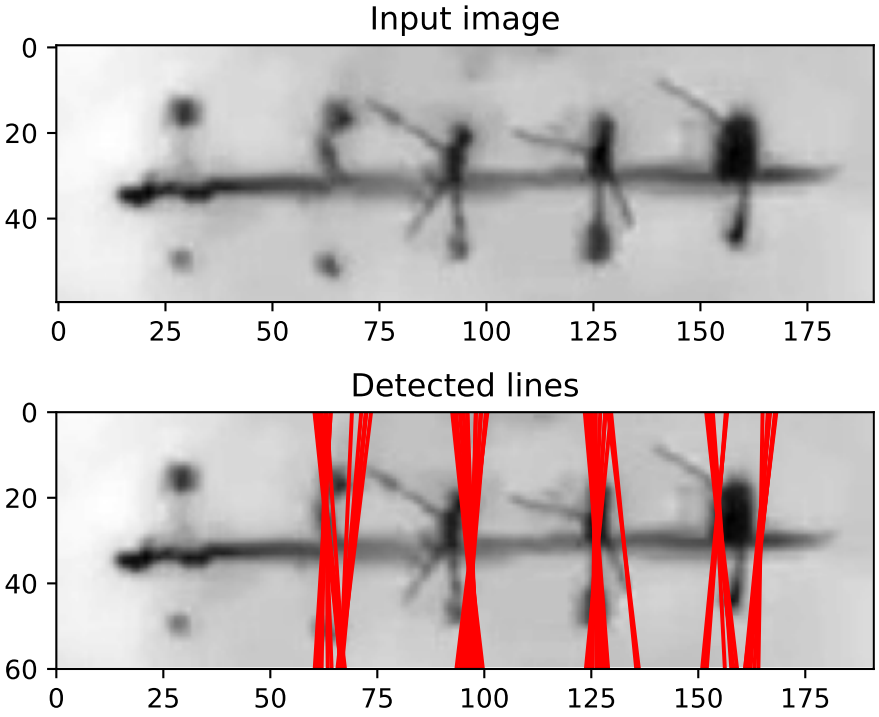
2 Další vyzkoušené přístupy

2.1 Další deskriptory

Kromě HOG bylo vyzkoušeno použít pro popis obrázku také deskriptor LBP, tedy Local Binary Pattern. Ten byl sice oproti HOG méně výpočetně náročný, ale dosahoval horších výsledků. Dále bylo zkoumáno, jestli nedojde ke zlepšení výsledků konkatenací vektorů z HOG a LBP s následným natrénováním klasifikátoru. Takový přístup však dosahoval stejných výsledků jako při použití pouze vektoru od HOG deskriptoru.

2.2 Hough Transform

Tento přístup navazuje na kapitolu 1.4, kde pro nalezené hrany bylo snahou pomocí Hough transformace detekovat počet stehů. Nejdříve k tomu byla využita *probabilistic_hough_line* z knihovny *skimage*, pro kterou ale bylo obtížné správně nastavit parametry, aby metoda dosahovala rozumných výsledků. Ze stejné knihovny byla také prozkoumána možnost použití funkce *hough_line*, čímž bylo získáno větší množství přímek. Na základě počtu přímek a jejich rozptylu by pak bylo možné určit počet stehů, ale toto řešení by bylo opět silně závislé na parametrech, které nelze rozumně nastavit pro všechny obrázky a pravděpodobně by toto řešení nedosahovalo lepších výsledků než dříve zmíněné metody.



Obrázek 4: Nalezené přímky pomocí funkce *hough_line*

3 Závěr

V rámci této semestrální práce byla navržena tři řešení pro detekci počtu stehů v obrázku. Nejlepších výsledků dosahovala neuronová síť, která je použita pro výslednou predikci počtu stehů vstupních obrázků, které lze dle zadání předávat pomocí argumentu. Klasické metody zpracování obrazu nedosahovaly příliš dobrých výsledků a bylo u nich občas obtížné nastavit hodnoty některých parametrů.

Při práci jsme často bojovali s nedostatečným počtem obrázků. Větší počet by nám umožnil rozdělit data na trénovací, validační a testovací část. Takto byly výsledky vyhodnocovány pouze na validační části, která i tak neobsahovala velký počet obrázků a v závislosti na zvolené hodnotě parametru *seed* se dosažené výsledky jednotlivých metod lišily.