



FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

Semestrální práce z KIV/ZVI

Detekce hran ve snímku pomocí neuronových sítí

Dominik Zappe – A23N0011P
(zapped99@students.zcu.cz)

Květen 2024

Obsah

1	Zadání	1
2	Teoretický popis metod	1
2.1	Konvoluční neuronová síť	1
2.2	HED	2
3	Implementace	3
3.1	Oficiální HED model (OpenCV + Caffe)	3
3.2	Vlastní konvoluční neuronová síť	3
3.3	Canny model s analytickými váhami	3
3.4	Vlastní menší HED model	4
3.5	Datasetsy	4
4	Experimenty a výsledky	4
5	Uživatelská příručka	8
6	Závěr	9

1 Zadání

Zadání z minulého roku: Detekce hran ve snímku (detekce v definovaném směru, směr maximálního gradientu, metody masek, Laplaceův operátor, detekce čáry a bodu, speciální hranové detektory — Canny, Marr-Hildreth, viz literatura.

V rámci semestrální práce jsem rozšiřoval své zadání z minulého roku, neboť jsem si letos předmět zapsal po druhé. Práce rozšiřuje tradiční metody detekce hran ve snímcích o metody užívající strojové učení a neuronové sítě.

2 Teoretický popis metod

2.1 Konvoluční neuronová síť

Konvoluční neuronové sítě jsou speciální typy neuronových sítí, které se často používají k úlohám zpracování obrazu – např. rozpoznávání objektů, detekce hran a segmentace. Konvoluční neuronové sítě jsou inspirovány biologickým zpracováním vizuálních informací v mozku [1].

Základními stavebními bloky konvolučních neuronových sítí jsou konvoluční vrstvy, pooling vrstvy a plně propojené vrstvy [1].

- Konvoluční vrstvy používají filtry – nebo kernely – k provádění konvolucí s vstupními daty. Konvoluce je operace, která aplikuje váhovou matici – filtr – na vstupní oblast obrazu a vytváří tzv. konvoluční mapy, které zachycují různé aspekty obrazu.
- Pooling vrstvy slouží k redukci prostorových rozměrů konvolučních map. Nejčastěji používanou operací v pooling vrstvách je tzv. *max pooling*, který vybírá maximální hodnotu z dané oblasti. To pomáhá snížit počet parametrů a zároveň zlepšuje invarianti vůči posunu vstupních dat.
- Po konvolučních a pooling vrstvách následují plně propojené vrstvy, které mají za úkol klasifikaci nebo regresi na základě extrahovaných funkcí z předchozích vrstev. Tyto vrstvy jsou dobře známé, neboť je používají klasické dopředné neuronové modely.

Konvoluční neuronové sítě jsou klasicky trénovány pomocí algoritmů jako je zpětné šíření chyby – *backpropagation*. Tento algoritmus aktualizuje váhy sítě tak, aby minimalizoval chybu výstupních predikcí. Jsou-li trénovány na dostatečném množství dat, jsou konvoluční neuronové sítě schopny se naučit reprezentace obrazů, díky kterým jsou schopny dobré generalizovat [1].

2.2 HED

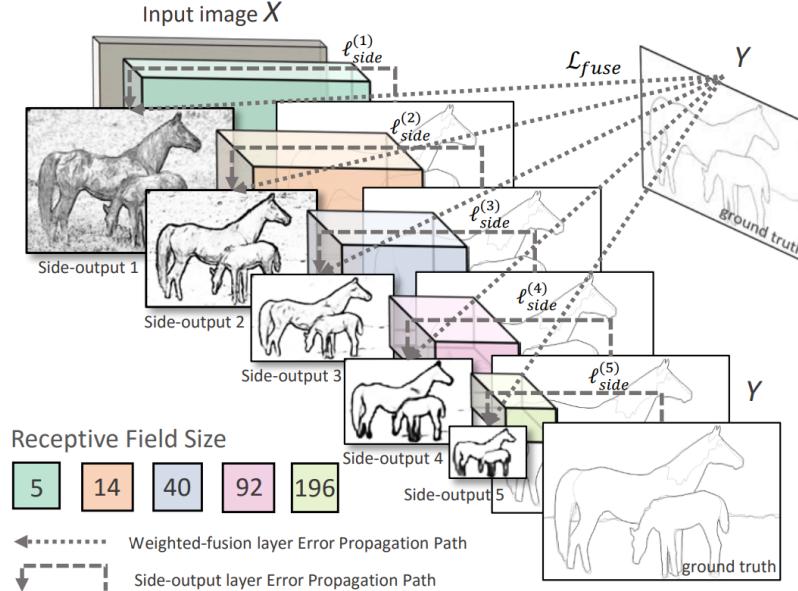
Jako „baseline“ pro srovnání různých metod byl zvolen model *Holistically-nested Edge Detection* (HED). Knihovní implementace tohoto modelu je součástí *OpenCV* knihovny, avšak model jako takový je implementován v knihovně *Caffe*. Oficiální implementace tohoto modelu je k nalezení na adrese <https://github.com/s9xie/hed>.

HED je metoda detekce hran, která byla poprvé představena v roce 2015. Jedná se o hlubokou konvoluční neuronovou síť, která se zaměřuje na detekci hran v obrazech. HED se snaží zachytit informace o hranách na různých úrovních detailu a škály [2].

Hlavními rysy HED jsou:

- Hierarchická architektura, která kombinuje informace z různých úrovní detailu, viz Obrázek 1. To umožňuje síti zachytit hrany různých délek a tloušťek, což vede k robustnější detekci hran.
- Úroveň spojení (Fusion level) – HED zahrnuje mechanismus nazvaný „fusion level“, který slouží k integraci informací z různých úrovní konvoluce. Tento mechanismus umožňuje síti rozhodnout, jaké informace z různých vrstev jsou relevantní pro detekci hran a jak je správně kombinovat.

Výstupem HED je pravděpodobnostní mapa hran, kde každý pixel v obraze má přiřazenou hodnotu, která udává pravděpodobnost toho, že se na daném místě nachází hrana. HED byl úspěšně použit v různých aplikacích zpracování obrazu, včetně segmentace objektů, extrakce rysů a rozpoznávání scén [2].



Obrázek 1: HED architektura sítě, převzato z [2]

3 Implementace

Celá aplikace je napsaná v jazyce *Python*. Celkem byly implementovány čtyři různé modely, které byly později zahrnuty i do GUI aplikace z minulého roku. Vlastní implementace neuronových sítí byla provedena v knihovně *torch*.

3.1 Oficiální HED model (OpenCV + Caffe)

Jak již bylo zmíněno, tento model byl zvolen za tzv. „baseline“, oproti které se ostatní řešení poměřují a snaží se docílit podobného výsledku.

Implementace tohoto řešení není nijak náročná, neboť OpenCV umožňuje jednoduše založit hlubokou neuronovou síť a posléze načíst její předtrénované váhy. Váhy pro tento model jsou dostupné online např. v oficiálním repozitáři modelu – <https://github.com/s9xie/hed>.

3.2 Vlastní konvoluční neuronová síť

Vlastní návrh konvoluční sítě vypadá následovně – nejprve je vstup proveden konvoluční vrstvou s maskou o velikosti 3×3 , očekávaná dimenze výstupu je 3 (RGB kanály – tedy barevný obrázek) a výstup z této vrstvy má dimenzi 8. Následně je na výstupu této vrstvy provedeno ReLU a max pooling vrstvou s velikostí masky 2×2 . Poté následuje opět konvoluční vrstva s filtrem velikost 3×3 , tentokrát je vstupní dimenze 8 a výstupní 16 – následuje opět ReLU a max pooling. Na závěr je provedena poslední konvoluce maskou o velikosti 3×3 , kde se navíc redukuje vstupní dimenze 16 na výstupní dimenze 1 (tedy šedotónový obrázek hran). Experimentálně bylo ověřeno, že výsledky jsou nejlepší, když se po poslední konvoluci již žádný max pooling ani plně propojená vrstva nedělá.

Bohužel se nepovedlo model natrénovat úplně dokonale a nejedná se tedy o *end-to-end* architekturu, kde by na vstupu byl obrázek a na výstupu detekované hrany. Výstup neuronové sítě je ještě nutný naprahovat, pak jsou ale výsledky uspokojivé.

3.3 Canny model s analytickými váhami

Při implementaci konvolučních neuronových sítí mě napadlo, že často užívána metoda pro detekci hran *Canny* je pouze chytřejší použití *Sobelova* operátoru a prahování.

Byl tedy vytvořen model, kterému říkám Canny Edge model s analytickými váhami. Tento model se totiž netrénuje a jsou mu váhy nastaveny analytickým řešením Cannyho detekce hran. Model obsahuje konvoluční vrstvy pro *Gaussovské rozmazání* (jak horizontální, tak vertikální), kde váhy byly nastaveny na základě *SciPy* implementace Gaussovského rozmazání. Dále model obsahuje konvoluční vrstvy pro *Sobelovo masky*

(opět horizontální a vertikální). Na závěr ještě model obsahuje 8 dalších konvolučních vrstev sloužící jako rotační filtr difference pixelů. Model následně tyto vrstvy používá ve vhodném pořadí tak, aby vznikl výsledek jako pří použití tradičního Cannyho detekce hran.

3.4 Vlastní menší HED model

Oficiální HED model obsahuje 5 *VGG-net* podsítí, kterých výsledky se ve finále konvoluují dohromady. Vlastní implementace menšího HED modelu obsahuje pouze 3 sjednodušené *VGG-net* „podsítí“ a výsledky těchto tří podsítí se ve finále konvoluují dohromady do výsledného obrázku hran.

Opět jako v případě vlastní konvoluční neuronové sítě se tento model nepodařilo plně natrénoval, aby se docílilo *end-to-end* sítě, kde by na vstupu byl obrázek a na výstupu detekované hrany – je opět nutné výstupy ze sítě ještě dále napraťovat. Opět jsou ale výsledky velmi uspokojivé.

3.5 Datasety

Byl vytvořen vlastní dataset pomocí semestrální práce z minulého roku – tradiční metody detekce hran. Ručně bylo vybráno 10 obrázků, ve kterých byly hrany detekovány pomocí implementovaných metod z minulého roku – konkrétně Canny (ručně volené parametry na základě obrázků), obousměrný Sobel a Gradientní magnituda centrální obousměrné diference.

Dále byl užit dataset jako kombinace dvou standardních datasetů – BIPED a UDED (<https://xavysp.github.io/MBIPED/> a <https://github.com/xavysp/UDED>).

Vlastní model konvoluční neuronové sítě a vlastní model zmenšené architektury HED byly oba trénovány na obou zmíněných datasetech. V GUI aplikaci je možné si vybrat konkrétní model – tedy např. HED na vlastních datech a HED na větších, standardních, datech.

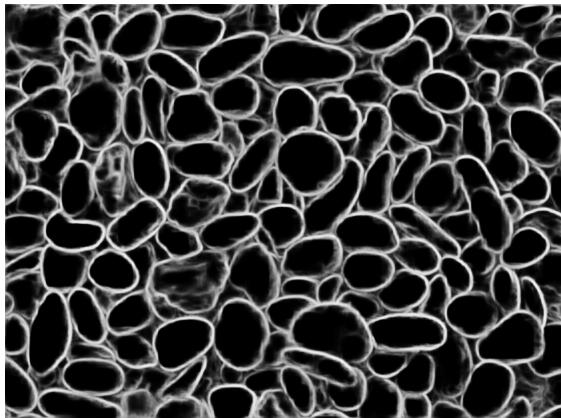
4 Experimenty a výsledky

Jak již bylo zmíněno dříve, hlavním experimentem prakticky bylo sítě správně natrénoval na různých datech ve snaze docílit podobného výsledku jako oficiální implementace HED modelu.

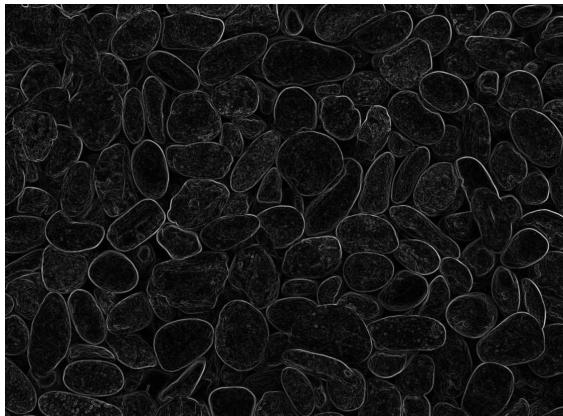
Nejvíce byl testován Obrázek 2, ve kterém se tradiční metody detekce hran poměrně snadno zmýlí, neboť hrany ve snímku nejsou jednoduše zřejmé (viz Obrázky 6 a 4). Tento snímek samozřejmě není součástí ani jednoho datasetu, tedy neuronové sítě ho nikdy při trénování neviděly.



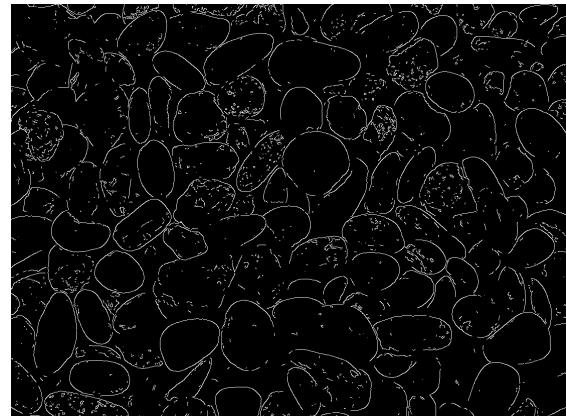
Obrázek 2: Obrázek kamení, kde jsou hrany těžko detektovatelné



Obrázek 3: Detekované hrany pomocí „baseline“ modelu HED



Obrázek 4: Detekované hrany pomocí tradičního Sobelova operátoru

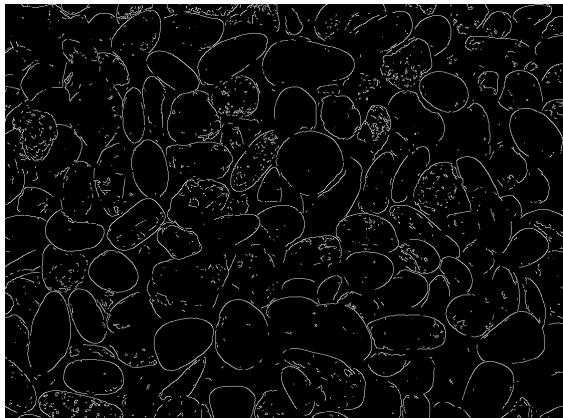


Obrázek 5: Detekované hrany pomocí tradičního Cannyho metody

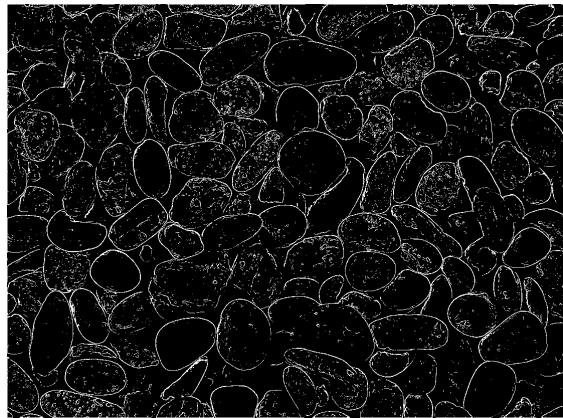
Je na první pohled vidět, že zvolený „baseline“ model, si vede lépe – Obrázek 3.

Canny model s analytickými váhami

Na následujících Obrázcích 6 a 7 je vidět srovnání tradiční metody Canny detekce a CNN metody s analytickými váhami. Je vidět, že výsledky jsou si podobné a že analytický model Canny opravdu funguje. Pro dosažení totožných výsledků by bylo nutné u tradičního Cannyho zvolit správné parametry prahů.



Obrázek 6: Detekované hrany pomocí tradičního Cannyho metody



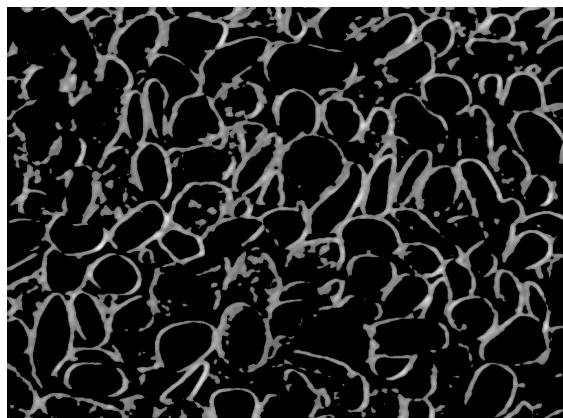
Obrázek 7: Detekované hrany pomocí konvoluční neuronové sítě Cannyho metody s analytickými váhami

Vlastní konvoluční neuronová síť

Na Obrázcích 8 a 9 jsou vidět výsledky vlastní navržené konvoluční neuronové sítě – trénování na velkých datech a vlastních datech.



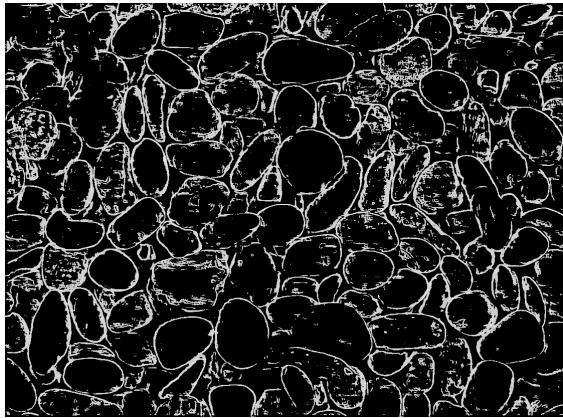
Obrázek 8: Detekované hrany pomocí konvoluční neuronové sítě – trénování na velkých standardních datech



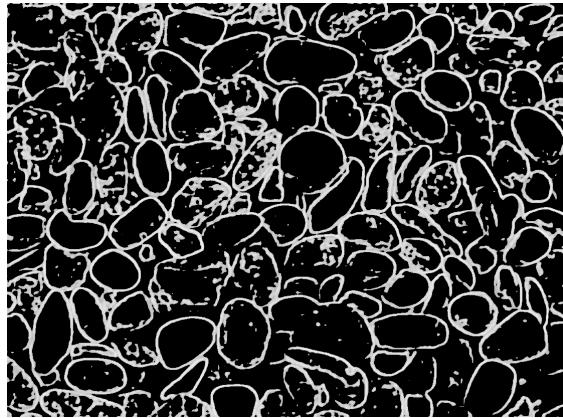
Obrázek 9: Detekované hrany pomocí konvoluční neuronové sítě .. trénování na vlastních datech

Vlastní menší HED model

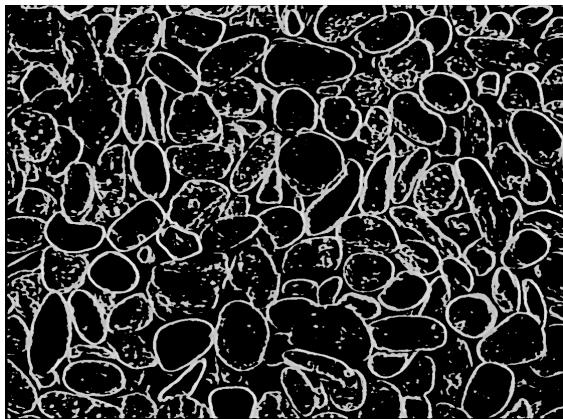
Na Obrázcích 10, 11 a 12 jsou vidět výsledky vlastního menšího HED modelu. Navíc je zde uveden Obrázek 13 pro srovnání oproti „baseline“ modelu.



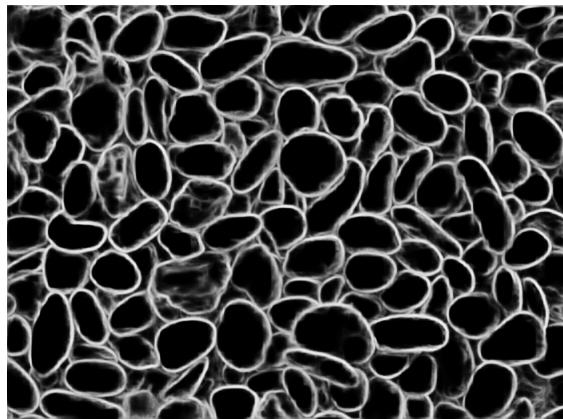
Obrázek 10: Detekované hrany pomocí vlastního modelu HED – trénování na velkých standardních datech



Obrázek 11: Detekované hrany pomocí vlastního modelu HED – trénování na vlastních datech



Obrázek 12: Detekované hrany pomocí vlastního modelu HED – trénování na vlastních upscalovaných datech



Obrázek 13: Detekované hrany pomocí „baseline“ modelu HED

5 Uživatelská příručka

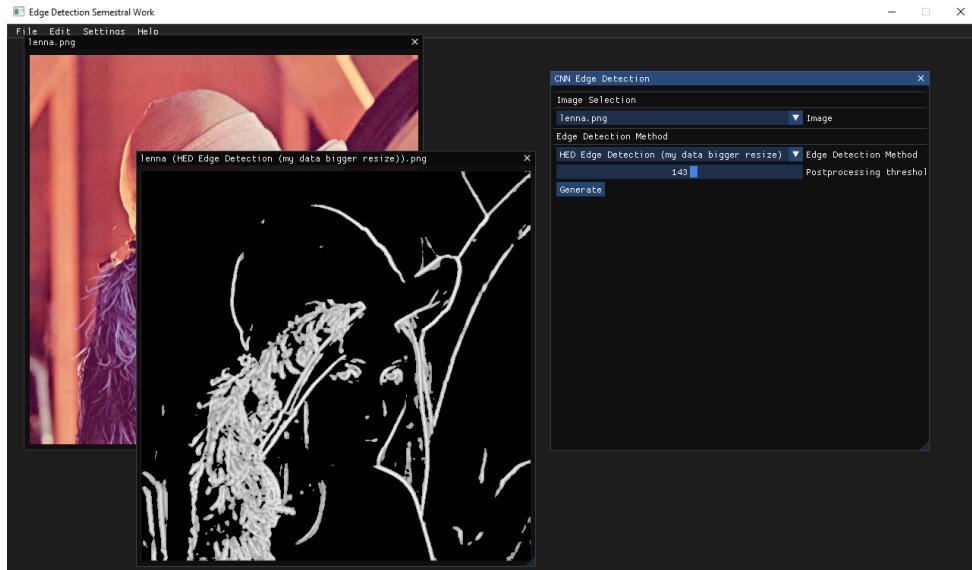
Aplikace byla testována na Pythonu 3.7+. Projekt obsahuje soubor *requirements.txt* – pomocí něj si je možné nainstalovat veškeré závislosti pomocí standardního manažeru balíčků pip.

```
C:\SP>pip install -r requirements.txt  
user@pc:/SP$ pip3 install -r requirements.txt
```

Spustitelný skript se nachází ve složce /src/gui a jmenuje se main.py. Aplikace je tedy možné spustit následovně:

```
C:\SP\src\gui>python main.py  
user@pc:/SP/src/gui$ python3 main.py
```

Aplikace neočekává žádné argumenty z příkazové řádky. Po spuštění se objeví grafické uživatelské rozhraní, kde je standardně hlavní menu, které nabízí položky *file*, *edit*, *settings* a *help*. V menu pod položkou *file* je možné si načíst obrázky do aplikace, respektive je uložit / exportovat na disk. Nabídka pod položkou *edit* je asi nejdůležitější a nabízí především *tradiční* metody hranové detekce (semestrální práce z minulého roku) a navíc letos nově *neuronové sítě* pro detekci hran. Navíc je možnost si snímek předzpracovat / dodatečně zpracovat rozmazáním a prahováním. Při otevření podokna pro detekci hran je možno si vždy vybrat snímek, na který se bude efekt aplikovat, metoda která se bude aplikovat a případně parametry dané metody. GUI je vidět na Obrázku 14.



Obrázek 14: Ukázka GUI

6 Závěr

Cílem semestrální práce bylo rozšířit aplikaci pro detekci hran v obrázcích o metody užívající neuronové sítě. Byla implementována vlastní architektura konvoluční neuronové sítě a zmenšená architektura HED modelu.

Vylepšením do budoucna je zvetšení počtu dat a trénování na větším počtu epoch (vše bylo trénováno na 100 epoch). Dalším vylepšením by mohlo být rozšíření zmenšeného modelu HED na klasicky velký HED model. V neposlední řadě by zlepšení mohlo být i v návrhu vlastní konvoluční sítě.

Reference

- [1] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015. URL: <https://arxiv.org/abs/1511.08458>, doi:10.48550/ARXIV.1511.08458.
- [2] Saining Xie and Zhuowen Tu. Holistically-nested edge detection, 2015. URL: <https://arxiv.org/abs/1504.06375>, doi:10.48550/ARXIV.1504.06375.