VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

TEXTURNÍ ANALÝZA

TEXTURE ANALYSIS

DIPLOMOVÁ PRACE

MASTER´S THESIS

AUTOR PRÁCE — Bc. Tomáš Opletal

AUTHOR

VEDOUCÍ PRÁCE — Ing. Peter Honec, Ph.D.

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Tomáš Opletal ***ID:*** 203537

Ročník: 2 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Texturní analýza

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnout metodu pro detekci anomálií v opakujícím se pravidelném vzoru (typicky došková střecha, kartáčovaný plech, tkané koberce…). Metodu otestujte na příkladu – detekce vad kalandru ve snímcích netkané textilie. Snímky budou dodány. Více detailů k zadání na Seznamte s metodami texturní analýzy

1. Analyzujte na dodaných vzorcích defekty vznikající při kalandrování netkané textlie poškozeným kalandrovacím válcem.
2. Navrhněte a implementujte vhodnou metodu pro odstranění pravidelného vzoru obtisknutého na materiálu.
3. Navrhněte a vhodnou metodu pro detekci anomálií v pravidelném vzoru.
4. Implementujte tuto metodu s přihlédnutím k požadovaným vlastnostem – spolehlivost, CPU zdroje apod.
5. Otestujte a zhodnoťte.

DOPORUČENÁ LITERATURA:

HLAVAC V., SONKA M., BOYLE R.: Image Processing, Analysis, and Machine Vision, ISBN 978-0495082521

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 17.5.2023

Vedoucí práce: Ing. Peter Honec, Ph.D.

doc. Ing. Petr Fiedler, Ph.D. předseda rady studijního programu

Abstrakt

Diplomová práce se zabývá texturní analýzou netkaných textilií. Cílem je detekovat vady materiálu, které mohly vzniknout při procesu vytváření kalandrů na netkané textilii. Pro tento účel byly otestovány různé metody. Jako nástroje byly využity Fourierova transformace s následným statistickým zpracováním a Neuronové sítě.

Klíčová slova

Texturní analýza, kalandr, počítačové vidění, Fourierova transformace, Strojové učení, Neuronové sítě, OpenCV, PyTorch, Matlab

Abstract

Diploma thesis discusses texture analysis. Goal is to detect anomalies on the material, which could arise during process of making calenders on the nonwoven material. This thesis is using methods of Fourier transform with statistics and Neural networks to detect position of the missing calender.

Keywords

Texture analysis, calandering, computer vision, Fourier transform, Machine learning, Neural networks, OpenCV, PyTorch, Matlab

Bibliografická citace

Opletal, Tomáš. *Texturní analýza*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřící techniky, 2023. 72 s. Diplomová práce. Vedoucí práce: Ing. Peter Honec, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	*Tomáš Opletal*
VUT ID studenta:	*203537*
Typ práce:	*Diplomová práce*
Akademický rok:	*2022/23*
Téma závěrečné práce:	*Texturní analýza*

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: 10. května ----------

podpis autora

V průběhu vytváření této práce mi byli velmi nápomocní pan Ing. Peter Honec, Ph.D., a pan Ing. Milan Krasňanský, kteří mi poskytli užitečné rady. Proto jim chci tímto poděkovat. Dále děkuji své rodině, která mě po celou doba podporovala.

V Brně dne: 10. května ----------

podpis autora

Obsah

SEZNAM OBRÁZKŮ **9**

SEZNAM TABULEK **10**

ÚVOD **11**

1. KALANDROVÁNÍ **12**

1.1 VÝROBA NETKANÝCH TEXTILIÍ 12
1.1.1 Uspořádání vláken do sítě *12*
1.1.2 Zpevňování *13*
1.1.3 Závěrečné úpravy *14*
1.2 ROZDĚLENÍ KALANDRŮ 14
1.2.1 Kalandry suché *14*
1.2.2 Kalandry speciální *14*
1.3 POUŽITÍ 14

2. FOURIEROVA TRANSFORMACE **16**

2.1 REPREZENTACE SIGNÁLU V ČASOVÉ DOMÉNĚ 16
2.2 REPREZENTACE SIGNÁLU VE FREKVENČNÍ DOMÉNĚ 16
2.3 FOURIEROVA TRANSFORMACE 2D SIGNÁLU 17
2.3.1 Využití 2D Fourierovy transformace *17*
2.3.2 Fourierova transformace čtverce *17*
2.3.3 Zobrazení frekvenčního spektra *18*
2.3.4 Základní vlastnosti *18*
2.3.5 Filtry *18*
2.3.6 Typy fltrů *19*

3. UMĚLÁ INTELIGENCE **21**

3.1 NEURON 22
3.1.1 Biologický neuron *22*
3.1.2 Umělý neuron *23*
3.1.3 Aktivační funkce *24*
3.1.4 Architektura neuronových sítí *24*
3.2 PLNĚ PROPOJENÉ VRSTVY 25
3.3 KONVOLUČNÍ VRSTVY 26
3.3.1 Konvoluce *26*
3.3.2 Pooling vrstva *27*
3.3.3 Upsampling vrstva *28*
3.4 ZTRÁTOVÁ FUNKCE 28
3.5 OPTIMALIZAČNÍ ALGORITMUS 28
3.5.1 Gradient descent *28*
3.5.2 ADAM *29*
3.6 UČENÍ MODELU NEURONOVÉ SÍTĚ 29
3.6.1 Dataset *29*
3.6.2 Učení Backpropagation *29*
3.6.3 Testovací a trénovací ztrátová funkce *30*
3.6.4 Overfitting *30*

3.6.5 Underfitting ... *31*
3.7 VYHODNOCENÍ MODELU ... 31
3.7.1 Matice záměn ... *31*
3.8 PYTORCH ... 32
3.8.1 Moduly knihovny PyTorch ... *32*
3.9 GOOGLE COLAB ... 33
4. AUTOENKODÉR ... 34
4.1 ENKODÉR ... 36
4.2 DĚLÍCÍ VRSTVA ... 36
4.2.1 Reparametrizační trik ... *36*
4.3 DEKODÉR ... 37
5. DETEKCE ANOMÁLIÍ VE VZORU ... 38
5.1 STATISTICKÁ DETEKCE ANOMÁLIÍ ... 41
5.1.1 Originální snímek ... *41*
5.1.2 Frekvenční spektrum ... *42*
5.1.3 Převod zpět do prostorové oblasti ... *43*
5.1.4 Detekce anomálii ... *45*
5.1.5 Vyhodnocení ... *46*
5.2 DETEKCE ANOMÁLIÍ POMOCÍ FOURIEROVY TRANSFORMACE ... 47
5.3 DETEKCE ANOMÁLIÍ POMOCÍ NEURONOVÝCH SÍTÍ ... 49
5.3.1 Data ... *49*
5.3.2 Nahrání snímků a vytvoření datasetu ... *51*
5.3.3 Vytvoření modelu variačního autoenkodéru ... *51*
5.3.4 Inicializace modelu a hyperparametrů ... *53*
5.3.5 Trénování modelu ... *53*
5.3.6 Validace modelu ... *54*
5.3.7 Průběh trénovací a validační ztrátové funkce ... *54*
5.3.8 Rekonstrukce modelu ... *55*
5.3.9 Rozdíl vstupního a výstupního ... *57*
5.3.10 Detekce anomálie ... *57*
5.3.11 Využití funkce cv::MatchTemplate() ... *57*
5.3.12 Využití Contours ... *59*
6. VYHODNOCENÍ METOD ... 60
6.1 STATISTICKÁ METODA ... 60
6.1.1 Robustnost ... *60*
6.1.2 Čas ... *60*
6.2 FOURIEROVA TRANSFORMACE ... 61
6.3 NEURONOVÉ SÍTĚ ... 61
6.3.1 Robustnost ... *61*
6.3.2 Přesnost ... *61*
6.3.3 Čas ... *65*
7. ZÁVĚR ... 66
LITERATURA ... 68
SEZNAM PŘÍLOH ... 71

SEZNAM OBRÁZKŮ

1.1 Spunlaid technologie [4] 13
1.2 Thermal bonding [4] 13
2.1 Tvar čtverce a jeho frekvenční spektrum [6] 18
2.2 Filtr typu dolní a horní propusť v porovnání s originálním snímkem 20
3.1 Model biologického neuronu [9] 22
3.2 Model umělého neuronu [11] 23
3.3 Obecná architektura neuronové sítě [14] 24
3.4 Architektura plně propojených vrstev [14] 25
3.5 Příklad konvoluce [15] 26
3.6 Příklad principu Max pooling vrstvy [15] 27
3.7 Průběhy trénovací a validační ztrátové funkce [15] 30
3.8 Matice záměn [20] 31
4.1 Obecná architektura autoenkodéru [23] 34
4.2 Porovnání latentního prostoru pro klasický a variační autoenkodér [24] 35
4.3 Architektura variačního autoenkodéru [24] 36
4.4 Reparametrizační trik [24] 36
5.1 Vada kontaminace černým vláknem na netkané textilií 38
5.2 Vada vzniklá při procesu kalandrování 39
5.3 Snímek netkané textilie s kosočtvercovou gravurou 41
5.4 Frekvenční spektrum snímků 42
5.5 Frekvenční spektrum po filtraci 43
5.6 Filtrovaný obraz 44
5.7 Gravura v binární podobě 44
5.8 Snímek vyjadřující počet bílých pixelů v určitém výřezu snímku 45
5.9 Prahovaný snímek 46
5.10 Výsledný snímek s detekovanými chybějícími kalandry 46
5.11 Analyzovaný snímek 47
5.12 Rozdíl ve frekvenční podobě 48
5.13 Výsledný snímek v prostorové oblasti 48
5.14 Příklad výřezu originálního snímku. 50
5.15 Ekvalizovaný originální snímek. 50
5.16 Průběh testovací a validační ztrátové funkce 55
5.17 Vstupní snímek modelu s vyznačenou anomálií 56
5.18 Výstupní snímek modelu 56
5.19 Rozdíl vstupního a výstupního snímku 57
5.20 Šablona 58
5.21 Detekovaná anomálie 59
6.1 Originální snímek s anomálií 62
6.2 Výsledek algoritmu pro detekci anomálií 63
6.3 Originální snímek s anomálií 64
6.4 Výsledek algoritmu pro detekci anomálií 64

SEZNAM TABULEK

6.1 Matice záměn pro vyhodnocení metody detekce Contours 62
6.2 Matice záměn pro vyhodnocení metody detekce pomocí šablony 63
6.3 Porovnání čau zpracování snímku různými metodami 65

Úvod

Tato práce se věnuje úloze, která je zaměřena na texturní analýzu. Úloha vznikla na základě reálného požadavku z průmyslu, kde vlivem poškození kalandrovacího válce může dojít k nesprávnému vytvoření kalandrovacího vzoru na netkané textilii.

Vyřešením této úlohy by došlo k mnohem jednodušší detekci vadných oblastí netkané textilie, čímž by se zvýšila kvalita výsledného výrobku a snížily náklady firmy.

Nejjednodušší metoda pro detekci těchto vad je odborný pohled a prozkoumání materiálu člověkem. Takováto metoda by byla dostatečná v případě detekování anomálie v dostatečně velkém vzoru (došková střecha). V případě netkaných textilií však může tato metoda velmi snadno selhat, jelikož kalandry mohou mít rozměry, které pro detekci člověkem jsou nevhodné. Proto je nutno k detekci anomálií využít jiné metody.

V rámci této práce tak byly navrhnuty, implementovány a otestovány různé metody pro detekci anomálií.

První metoda, která byla implementována, je jednoduchá statistická metoda. Využívá se Fourierovy transformace, která je velmi významná v tom, že dokáže zobrazit jednotlivé frekvenční složky, které snímek obsahuje. Práce využívá frekvenční analýzy, kde se pomocí transformace odfiltruje část spektra, která je pro další výpočty nevýznamná. Postupně se pomocí dalších výpočtů získá informace o pozici chybějícího kalandru.

Další otestovanou metodou bylo využití pouze Fourierovy transformace. Metoda by měla pracovat na principu porovnání frekvenčního spektra správného a vadného snímku.

Jako poslední navrhnutou, implementovanou a otestovanou metodou, bylo využití neuronových sítí. Konkrétněji byla využita architektura variačního autoenkodéru, pomocí které se síť na datasetu naučila vzor a tento vzor byla schopna zpět rekonstruovat na výstupu sítě.

Práce je rozdělena na několik částí. První je teoreticky zaměřená, kde je popsána technologie kalandrování, Fourierova transformace, neuronové sítě a variační autoenkodér.

Následně bylo popsáno řešení úlohy pomocí jednotlivých metod, kde byly popsány kroky a princip fungování. V poslední kapitole je vyhodnocení jednotlivých metod z hlediska robustnosti, spolehlivosti a výpočetní náročnosti. V závěru se nachází shrnutí této práce a dosažených výsledků.

1. KALANDROVÁNÍ

Kalandrování patří mezi nejčastější pojení vrstev netkaných textilií na bázi termoplastů (nazýváno též jako termické pojení – thermal bonding). Proces kalandrování spočívá v tom, že materiál je veden v plné šíři přes dva válce kalandru, které jsou zahřáty. Působením tlaku horkých válců na materiál dojde k termickému propojení jednotlivých vláken a zároveň otisku profilu kalandrovacího válce. Kalandrováním lze ovlivnit mechanické vlastnosti materiálu jako je například lesk, hladkost nebo omak. Dále lze pomocí kalandrování vytvořit na materiálu různé vzory. [3]

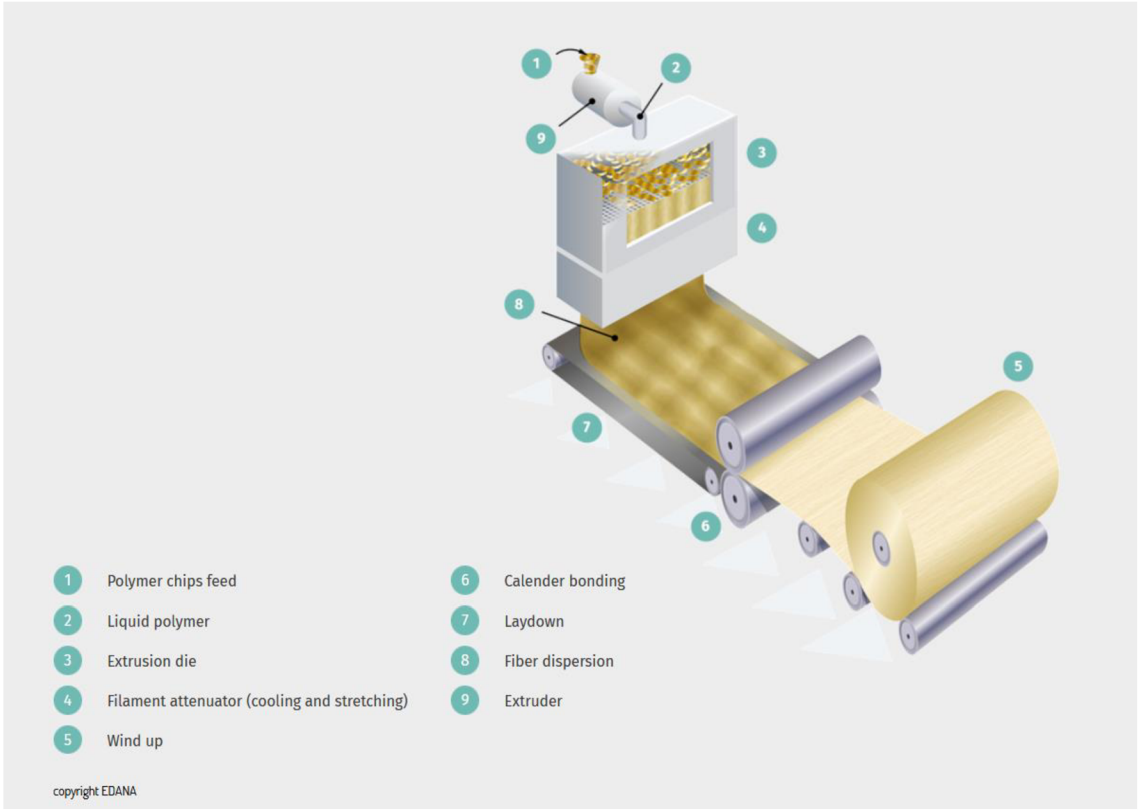
1.1 Výroba netkaných textilií

Proces výroby netkaných textilií se skládá ze tří částí.

1.1.1 Uspořádání vláken do sítě

V prvním kroku je třeba z vláken vytvořit síť. K tomu lze využít několik různých technologií. Lze využít technologie Drylaid-Carded, Short fiber airlaid, Wetlaid, Spunlaid, Meltblown a Submcron spinning. Podrobný popis každé technologie se nachází na stránce výrobce Edana. [4]

Následně je popsána pouze technologie Spunlaid, schéma je na obrázku 1.1.



Obrázek 1.1 Spunlaid technologie [4]

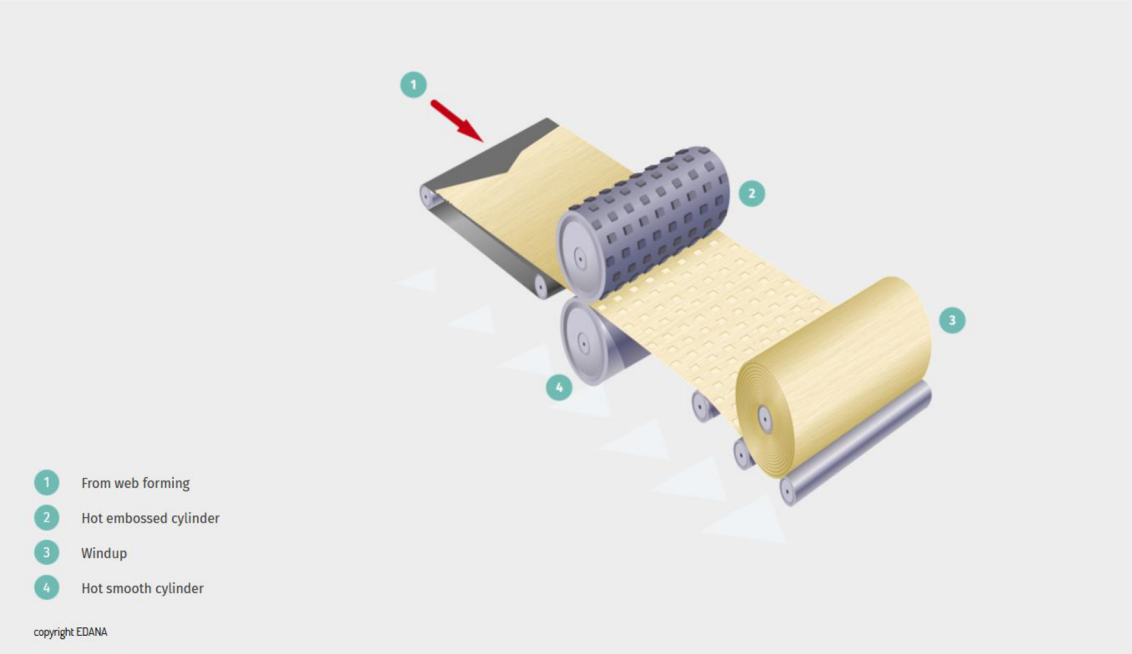
Pro vytváření sítě se využívá termoplastů. Granule termoplastů jsou vytlačovány do filamentů pomocí zvlákňovacích trysek. Následně jsou vlákna natahována (dloužena) a ochlazována před přesunem na dopravníkový pás, kde pomocí válců dojde k vytvoření jednotného rouna. Výsledkem tohoto procesu jsou netkané textilie, které mají vyšší pevnost, než například netkané textilie vyrobené pomocí technologie Drylaid–carded. Nevýhodou je, že výběr vhodné vstupní suroviny je omezený. [4]

1.1.2 Zpevňování

Pomoci procesu vytváření sítě z jednotlivých vláken dojde k vyrobení jednolitého rouna, které však ještě mají omezenou pevnost (v závislosti na zvolené metodě výroby). Proto rouno prochází dalším procesem, kterým je materiál více zpevněn.

Způsoby pro zpevnění tohoto materiálu jsou Thermal bonding, Mechanical bonding a Chemical bonding. Jednotlivé metody jsou popsány například na stránkách výrobce Edana. [4]

Následně je popsáno zpevnění materiálu metodou Thermal bonding.



Obrázek 1.2 Thermal bonding [4]

Na obrázku 1.2 se nachází proces zpevňování pomocí metody Thermal bonding. Využívá se vlastnosti termoplastů, při které vlivem působením tepla a tlaku na vlákna termoplastů dojde k vytváření vazeb mezi nimi, čímž se vytvoří výsledné rouno netkané

textilie. Zároveň pokud se na kalandrovém válci nachází vzor, dojde k vytlačení tohoto vzoru na materiál, čímž vznikne netkaná textilie s požadovanou gravurou.[4]

1.1.3 Závěrečné úpravy

Možnost kombinace různých materiálu a různých postupů procesu při vytváření netkané textilie umožňuje různorodost netkaných textilii z hlediska jejich vlastnosti a použití.

Různorodost lze ještě rozšířit závěrečnými úpravami materiálu. Může se jednat o mechanické úpravy – natahování. Případně chemické úpravy, kterými lze zajistit, aby povrch textilie byl vodě odpudivý, nebo absorbující (avivážování). [4]

1.2 Rozdělení kalandrů

Základní dělení kalandrů je na suché a speciální.[3]

1.2.1 Kalandry suché

Suché kalandry ovlivňují povrchové vlastnosti materiálu.

Matovací kalandr je zaměřen na zajištění plného měkkého omaku textilie. V případě nití zachová kruhový průřez. Valivý kalandr zajistí u textilie vyšší lesk a plný omak. Kalandr třecí zajistí u textilie vysoký lesk, který vznikne z důvodu tření mezi vyhřívaným kovovým válcem a válcem měkkým.[3]

1.2.2 Kalandry speciální

Tento typ kalandrů má na svém povrchu rytinu (gravuru), která svým působením na materiál vytlačuje do materiálu vzor.

Zde tedy může vzniknout hlavní vada, která se cyklicky opakuje na materiálu. Pokud bude část rytin na válci poškozena (vylomení jedné části rytiny opotřebením), potom při průchodu materiálu válci dojde k nesprávnému vytvoření vzoru na materiálu, což má negativní vliv na estetické i mechanické vlastností výsledného produktu. [3]

1.3 Použití

Netkané textilie a výrobky z nich jsou lidmi každodenně využívány.

Nezastupitelnou roli mají v případě hygienických produktů. Netkaná textilie je vyrobena tak, aby absorbovala vlhkost. Následně lze z takovéhoto materiálu vytvořit například dětské plenky, které mají výhodu v tom, že jsou pohodlné na nošení, protože materiál je jemný a prodyšný zároveň.

V případě zemědělství se netkané textilie používají pro zvýšení efektivity a výnosnosti zemědělských plodin a stromů. Plodina pokrytá netkanou textilií je chráněna před mrazem, případně před škůdci. Využívají se i černé netkané textilie, které

zabraňují průniku UV záření a tím pádem i růstu plevele pod touto vrstvou. Zároveň je dostatečně lehká, takže nebrání plodině v růstu a je dostatečně prodyšná, takže nebrání přístupu kyslíku k plodině.

Netkané textilie se taktéž využívají ve stavebnictví. Používají se jako tepelná i zvuková izolace, paropropustná izolace, do vzduchových filtrů nebo jako podložní vrstva zastřešení.

Hojně se využívají ve zdravotnictví. Vyrábí se z nich chirurgické roušky a respirátory, různé ubrousky, oděvy nebo povlečení. Výhodami využití netkané textilie v odvětví zdravotnictví jsou materiály odpuzující vlhkost, zároveň filtrující částice obsažené ve vzduchu, s vysokou prodyšností. [5]

2. FOURIEROVA TRANSFORMACE

Fourierova transformace je matematický nástroj, sloužící k analýze signálu. Jedná se v podstatě o reprezentaci analyzovaného signálu ve formě harmonických signálů sinus a cosinus. Jedná se o matematickou operaci, sloužící k převodu signálu z časové oblasti do oblasti frekvenční. Reprezentace signálu ve frekvenční oblasti je velmi užitečná v případě spektrální analýzy signálu. [1]

Pomocí Fourierovy transformace lze signál rozložit na jednotlivé frekvenční složky, čímž získáme přehled o tom, které frekvence se v signálu vyskytují. Důsledkem toho je to, že signál lépe pochopíme z hlediska frekvenčních složek, a tím lze snadněji manipulovat s takovým signálem. Nejčastěji se Fourierovy transformace využívá při redukci a odstranění šumu v obraze, případně pro vybrání pouze určitých struktur a rysu v obraze. Dále se Fourierova transformace využívá v případě komprese dat. [1][2]

2.1 Reprezentace signálu v časové doméně

Signál je nejčastěji reprezentován v takzvané časové doméně. V této formě amplituda signálu *x(t)* je vyjádřena v závislosti na čase. Pokud takovýto signál je vyjádřen v každém okamžiku času, jedná se o signál spojitý. Pokud signál je definován pouze v určitých okamžicích, jedná se o signál diskrétní. Matematicky je takovýto signál vyjádřen jako číslo vyjadřující nezávislou proměnou *n* a hodnota *x(n)*. Předpokládá se, že takový signál je definován se stejnou periodou *T*. Pro převod signálu z frekvenční domény zpět do časové domény se využívá vzorce [2]

$$f(t) = \int_{-\infty}^{+\infty} F(u) e^{2\pi jut}\, dt\,, \tag{2.1}$$

2.2 Reprezentace signálu ve frekvenční doméně

Každý signál x(t) lze převést pomocí Fourierovy transformace do frekvenční domény. Matematicky lze převod popsat jako

$$F(u) = \int_{-\infty}^{+\infty} f(t) e^{-2\pi jut}\, dt\,, \tag{2.2}$$

Kde F(u) je frekvenční obraz funkce x(t), x(t) je funkce v časové doméně a $e^{-2\pi jut}$ je tzv Eulerův vzorec, který lze vyjádřit jako

$$e^{-2\pi jut} = \cos(2\pi ut) - isin(2\pi ut)\,, \tag{2.3}$$

Vztahy (2.1) a (2.2) vyjadřují Fourierovu transformaci pro převod signálu z časové domény do frekvenční domény a obráceně. [2]

2.3 Fourierova transformace 2D signálu

Ačkoliv na první pohled to není jednoznačné, do frekvenční oblasti lze převést i snímek. Důvodem je to, že Fourierovu transformaci, která je v nejjednodušším případě využita pro transformaci 1D signálu, lze rozšířit i na signál 2D (další rozšíření je na signál 3D).

Jelikož snímek si lze představit jako 2D prostor, lze aplikovat Fourierovu transformaci na snímek. V takovém případě vlastnosti zůstanou stejné. Stále se jedná o operaci, pomocí které se 2D signál převede do frekvenční oblasti, kde je možné s tímto obrazem manipulovat dle požadavků. Narozdíl 1D Fourierově transformaci se liší způsobem výpočtu, kdy se za f(t) zavede substituce f(x,y), a F(u) se nahradí F(u,v) a integrace se provede přes dvě proměnné místo jedné. [2][6]

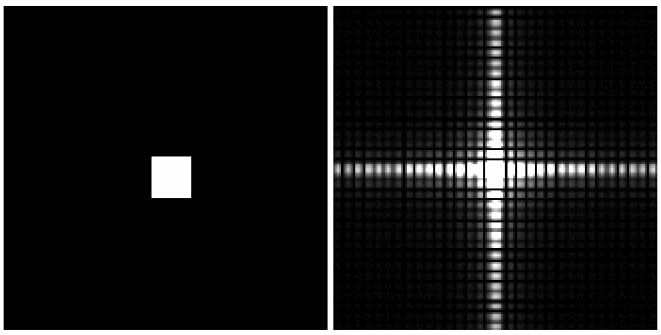
V případě dvourozměrné Fourierovy transformace je základem tzv. prostorová frekvence, kterou si lze představit jako funkce sinus nataženou do plochy. Tato funkce má 3 parametry – frekvenci, posun a natočení. Výsledkem těchto prvků je spektrum prostorových frekvencí, které tvoří bázi. [6]

2.3.1 Využití 2D Fourierovy transformace

Využití je stejné jako případě 1D Fourierovy transformace. Je možno provádět frekvenční filtraci pomocí filtrů. Existují různé typy takových filtrů. Například filtr typu horní propusť, dolní propusť, pásmová propusť a jejich modifikace. Pomocí těchto filtrů lze snímek rozmazávat, zaostřovat, a odstraňovat části obrazu, které mají určitou frekvenci. Pomocí Fourierovy transformace je možno odstraňovat šum v obraze. Fourierovu transformaci lze využít k zvýraznění hran, případně některých objektů v obraze. [6]

2.3.2 Fourierova transformace čtverce

Pro základní demonstraci je využit čtverec a jeho obraz ve frekvenčním spektru na obrázku 2.1.



Obrázek 2.1 Tvar čtverce a jeho frekvenční spektrum [6]

V levé části obrázku je snímek čtverce, v pravé části obrázku je jeho frekvenční spektrum. Ve středu je stejnosměrná složka, která udává střední hodnotu obrazu. Směrem ke krajům se frekvence zvyšuje, zároveň se však snižuje amplituda pro tyto frekvence.
Nižší frekvence tedy vytvářejí základní tvar, vyšší frekvence přidávají detaily – hrany. [6]

2.3.3 Zobrazení frekvenčního spektra

Jelikož výsledek Fourierovy transformace je v komplexním tvaru, zobrazení je náročné. Dalšími problémy je rozsah zobrazovaných hodnot a to, že spektrum má zobrazenou stejnosměrnou složku v rohu obrázku.

Řešení je takové, že se zanedbává fáze a pro další výpočty se využívá pouze amplituda. Pro lepší pochopení spektra se přesouvá stejnosměrná frekvence z rohu do středu. Rozsah se dá upravit pomocí výpočtu logaritmu, před zobrazením samotného spektra. [6]

2.3.4 Základní vlastnosti

Lze 2D Fourierovu transformaci rozdělit na dvě postupné 1D Fourierovy transformace. Lze to vidět snadno ze vzorce pro 2D transformaci, ve které lze rozdělit výpočet na dva nezávislé integrály v jednotlivých směrech. [6]

V obou doménách lze signály sčítat a násobit. Platí princip superpozice. [6]

Pokud jsou v prostorové doméně hrany, pak ve frekvenční doméně bude vidět jejich frekvence. [6]

Pro reprezentaci 2D signálu ve frekvenční oblasti se využívá 2D spektrum, které je symetrické podle počátku. [6]

Pokud je frekvenční obraz převeden do tvaru, ve kterém se stejnosměrná složka nachází ve středu, potom nejnižší frekvence je na počátku a se vzrůstající vzdálenosti od počátku se zvětšuje i frekvence. [6]

V případě kdy není 2D signál čistě periodický (obsahuje jasové nebo geometrické zkreslení), pak se ve frekvenčním spektru objeví vyšší harmonické. [6]

2.3.5 Filtry

Filtrace pomocí Fourierovy transformace se využívá nejčastěji k potlačení šumu a zvýraznění hran. Filtrace je potom realizována pomocí konvolučních filtrů. Konvoluční filtry jsou však mnohem snadnější pro výpočet ve frekvenční oblasti, protože náročná operace konvoluce v prostorové oblasti se změní na jednoduchou operaci násobení. Zároveň je ale potřeba filtrovaný snímek a filtr převést do frekvenční oblasti a následně

zpět do prostorové oblasti. To znamená, že tento převod bude náročný z hlediska výpočtů a je důležité se rozhodnout, zdali bude zvolena filtrace v prostorové nebo frekvenční oblasti v závislosti na každé úloze. [6]

Obecně lze říci, že filtrace v prostorové oblasti je výhodnější v případech, kdy je využito menších filtrů, které nebudou mít velkou výpočetní náročnost. Dále pouze v prostorové doméně lze využít nelineární filtry. Naopak ve frekvenční doméně je výhodnější využít větších a složitějších filtrů. [6]

Zároveň pro obě domény je potřeba filtry realizovat odlišným způsobem. V prostorové doméně jsou výhodnější použít filtry s ostrými hranami, protože jsou výpočetně méně náročné. Ve frekvenční oblasti je výhodnější použít hladké filtry.

Filtry ve frekvenční oblasti jsou nejčastěji realizovány pomocí Gaussovy křivky. Tento typ filtrů je rozmazávací. [6]

2.3.6 Typy fltrů

Pro filtraci ve frekvenční oblasti se využívá několika typu selektivních filtrů.

Filtr typu dolní propusť má frekvenční charakteristiku, která má od počátku nejvyšší hodnoty, zatímco v místech nejdále od počátku hodnoty nejnižší. Tímto typem se zajistí to, že v případě použití tohoto typu filtru vznikne výsledný snímek, který bude obsahovat pouze nízké frekvence. To znamená, že snímek bude spíše rozmazaný, bez jednoznačných hran. [6]

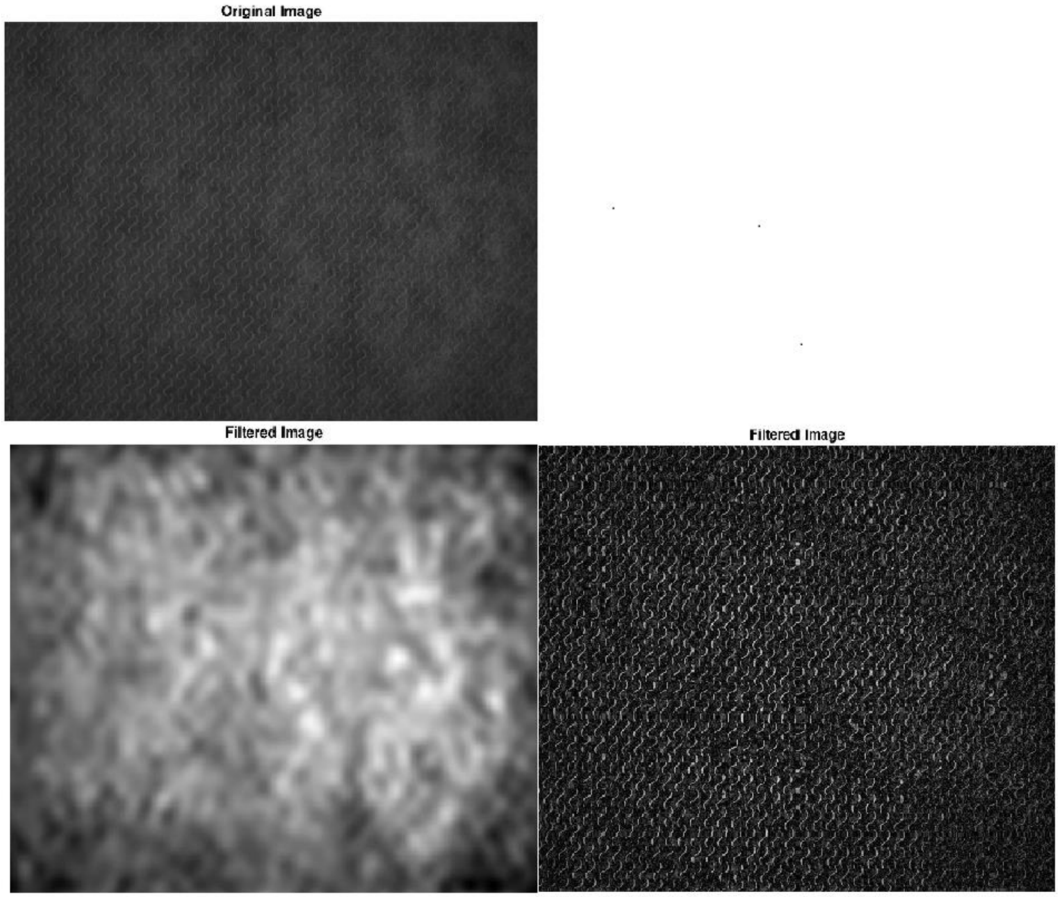
Dalším typem je filtr horní propusť. Ten vypadá tak, že nejnižší hodnoty má kolem středu, zatímco směrem k okrajům se hodnoty zvětšují. Tímto typem filtru vznikne snímek, který bude obsahovat pouze vysoké frekvence. To znamená, že na snímku budou zvýrazněné hrany. [6]

Dále lze využít pásmovou propusť, která propouští pouze frekvence v určitém pásmu. Tím se docílí toho, že například na výsledném snímku budou zobrazeny větší hrany, zároveň však bude potlačen šum, který je většinou vysokofrekvenční. [6]

Pro tvorbu filtrů není vhodné využít ostré ohraničení. Je výhodnější udělat přechod kontinuální na hranách filtrů. To dokáže zajistit například Butterworth nebo Gauss filtr. [6]

Pro lepší představu funkce filtrů typu horní a dolní propusť byl využit snímek netkané textilie s vlnovým tvarem gravury. Na tento snímek byl použit filtr typu dolní a horní propusť. Lze to vidět na obrázku 2.2. Pod tímto snímkem vlevo je vidět dolní propusť. Lze vidět, že takový snímek je rozmazaný, a jedná se v podstatě o pozadí originálního snímku.

Vpravo je vidět funkce filtru typu horní propusť. Není na něm téměř vidět pozadí snímku, na druhou stranu jsou zvýrazněné hrany. V tomto případě více vynikne gravura. [6]



Obrázek 2.2 Filtr typu dolní a horní propusť v porovnání s originálním snímkem

3. UMĚLÁ INTELIGENCE

Jak už název této kapitoly napovídá, v další části této práce bude popsána umělá inteligence. Jedná se o počítačový systém, který napodobuje lidský způsob uvažování při řešení úloh. Na základě vstupních dat vytváří výstupní předpovědi. Takový systém se vyznačuje tím, že je schopen se na základě dat dodaných člověkem dokáže učit. Tím se vylepšuje přesnost odpovědí.[7][8]

Historie této technologie sahá do 50. let 20 století. Jako vůbec první práci, která se aktuálně považuje za umělou inteligenci, sestrojil Walter Pitts. V tomto časovém období v oblasti umělé inteligence figurovaly další známé osobnosti. Například Alan Turing, který definoval v roce 1950 tzv. Turingův test. Tím se vytvořila obecná definice toho, co lze považovat za umělou inteligenci. Už v roce 1997 společnost IBM vytvořila umělou inteligenci, která dokázala porazit tehdejšího mistra světa v šachách. Poslední dobou se jedná o velmi rychle se rozvíjející se obor, ve kterém se dosáhlo významných objevů a úspěchů. [8]

Vědní obory, které zasahovaly do umělé inteligence, jsou například:

Filozofie – co je inteligence? Jakým způsobem vzniká vědomí člověka z fyzického mozku? O těchto otázkách přemýšlel už starořecký myslitel Aristoteles.

Matematika – co vše může být matematicky vyjádřeno? Jakým způsobem vypočítáno?

Neurovědy – jakým způsobem mozek přijímá informace, zpracovává je a generuje akci na základě přijímaných informací? Inspirace fungování umělé inteligence lze nalézt v chování lidské inteligence.

Psychologie – jakým způsobem lidé myslí a chovají se? Jisté podobnosti lze nalézt při studiu kognitivní psychologie, která považuje mozek za zařízení sloužící ke zpracování informaci.

Počítačové vědy – jakým způsobem sestrojit efektivní počítač? Pro úspěšnou implementaci umělé inteligence je potřeba dvou věcí – inteligenci a výpočetní jednotku.

A mnohem více.[8]

Jak je vidět, umělá inteligence se prolíná s mnoha dalšími obory.

Umělou inteligenci lze rozdělit na tři typy:

1. Omezená umělá inteligence – Takovýto typ umělé inteligence se vyznačuje tím, že dokáže provádět určitou činnost lépe než člověk, zároveň však v jiné úloze selhává. Všechny příklady umělé inteligence jsou v dnešní době považovány za omezené, ačkoliv její vývoj jde mílovými kroky. [7]
2. Obecná umělá inteligence – inteligence, která dosahuje a přesahuje schopnosti člověka řešit úlohy z mnoha různých odvětví. Tento typ umělé inteligence v tuto chvíli neexistuje. Objevují se názory, že nová verze umělé inteligence Chat-GPT 5, která bude vydána na konci roku 2023, této úrovně dosáhne. [7][25]

3. Umělá superinteligence – Takový systém by byl schopen překonat lidi ve všech oblastech. Tím pádem člověk by přišel o své výsostní postavení na naší Zemi, které vzniklo právě díky naší inteligenci. V tuto chvíli se jedná pouze o hypotetickou situaci, před kterou se však objevují varování od různých osobností napříč světem.[7]

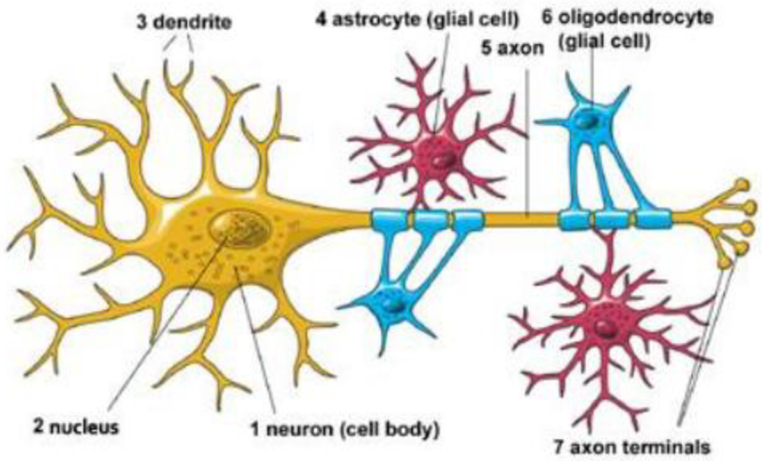
V další části této práce bude popsána neuronová síť a její základní stavební prvky. Ta je podmnožinou Umělé inteligence. Neuronová síť byla v rámci této práce navrhnuta, implementována a otestována.

3.1 Neuron

Neuron je považován za základní stavební jednotku neuronových sítí. Umělý neuron je inspirován biologickým neuronem. [10]

3.1.1 Biologický neuron

Biologický neuron je základním stavebním prvkem lidského mozku. Slouží k přenosu informací mezi jednotlivými neurony, který tvoří celý mozek. Přenos informací probíhá pomocí elektrických a chemických impulsů. Biologický neuron se skládá ze třech základních částí. Buňka neuronu, axony a dendrity. Axony slouží k odesílání signálu do ostatních neuronů, dendrity slouží přijímání signálu z jiných neuronů. Mezi neurony se nachází synapse, což jsou kanály, kterými se mezi sebou posílají signály. Jednoduchý model biologického neuronu se nachází na obrázku 3.1.[9]



Obrázek 3.1 Model biologického neuronu [9]

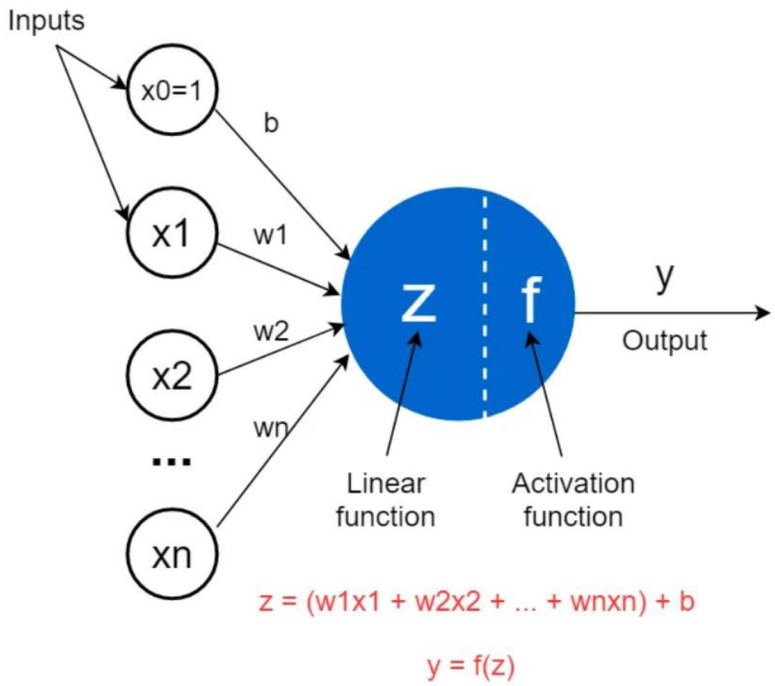
3.1.2 Umělý neuron

Umělý neuron funguje podobným způsobem jako neuron biologický. To znamená, že přijímá signál z ostatních neuronů pomocí vstupů, které obsahují jednotlivé váhy. Váhy vyjadřují důležitost jednotlivých vstupů. Dále jako vstup je považován tzv. bias. Vnitřek umělého neuronu si lze představit jako matematickou funkci vstupů, která generuje výstup pro aktivační funkci. Výstup celého neuronu je pak jedna hodnota, která může být vstupem pro další neurony. [11]

Uvnitř neuronu jsou dvě funkce. První je lineární, kde se jedná o sumu jednotlivých vstupů x_n vynásobené vahami w_n a biasu b. Bias se používá v situaci, při níž jsou všechny vstupy rovny 0, aby na výstupu lineární funkce byla nenulová hodnota, která se bude rovnat právě biasu. Váhy a bias se považují za parametry neuronů. Jejich optimální hodnota se nastavuje v průběhu procesu učení. [11]

Druhá aktivační funkce je nelineární. Cílem je zahrnout do neuronové sítě i nelinearity, díky kterým je možno modelovat nelinearity, které se mohou v datech nacházet. [11]

Nákres umělého neuronu lze nalézt na obrázku č. 3.2.



Obrázek 3.2 Model umělého neuronu [11]

3.1.3 Aktivační funkce

Aktivační funkce se dělí na lineární a nelineární. Lineární aktivační funkce ve většině případů selhávají, jelikož nejsou schopny pracovat s daty, která jsou většinou komplexnějšího charakteru a pro jejich správnou interpretaci je potřeba využít nelineární funkce. [12]

Nejpoužívanější nelineární aktivační funkci je ReLU (Rectified linear unit). Jedná se o po částech lineární funkci, která převede na výstup vstup v případě, když je větší než 0. Pokud je vstup menší nebo roven 0, potom na výstupu bude hodnota 0. Hlavní výhodou této nelineární funkce je jednoduchost. Pomocí ReLU je možné učit hluboké neuronové sítě. Matematicky ji lze vyjádřit jako

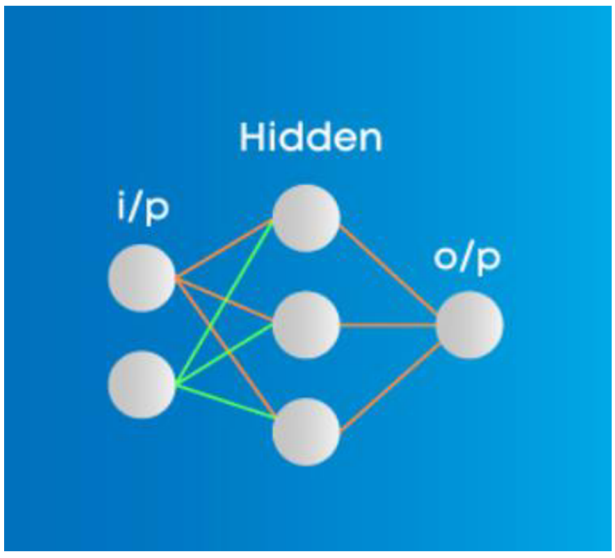
$g(z) = \max(0, z)$. [13]

Dalším příkladem často používané nelineární aktivační funkce je sigmoida. Nejčastěji je využívána tam, kde je na výstupu neuronové sítě predikována pravděpodobnost, že její hodnoty budou v intervalu (0,1). Hlavní nevýhodou sigmoidy je saturace hodnot. To znamená, že jakákoliv vyšší vstupní hodnota větší než 1 generuje na výstupu funkce hodnotu 1. [12]

3.1.4 Architektura neuronových sítí

Existují různé typy architektur neuronových sítí. Obecná architektura neuronových sítí je vidět na obrázku níže.

Obsahuje jednu vstupní vrstvu, jejímž vstupem jsou data. Má tedy stejný rozměr, jako data. Dále obsahuje skrytou vrstvu, která může obsahovat více jednotlivých vrstev. Poslední je výstupní vrstva, která reprezentuje výstup neuronové sítě. Jednotlivé vrstvy mohou být různých typů. Plně propojené vrstvy a konvoluční vrstvy jsou popsány dále. [14]



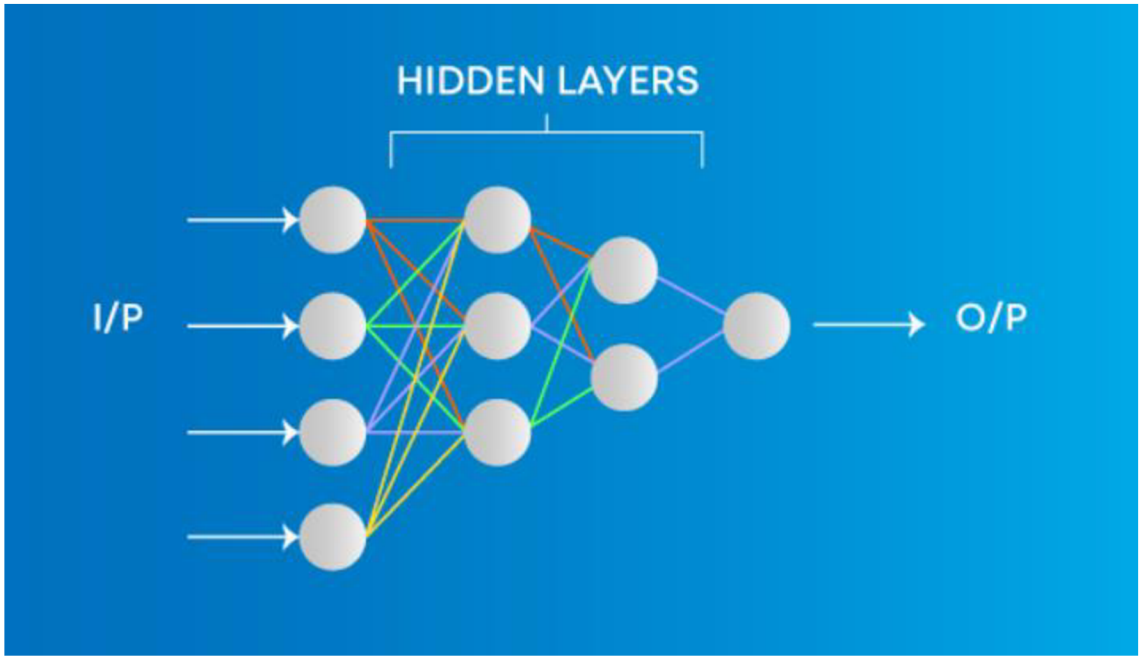
Obrázek 3.3 Obecná architektura neuronové sítě [14]

3.2 Plně propojené vrstvy

Jedním z nejpoužívanějších typů uspořádání neuronových sítí jsou Plně propojené vrstvy. Obsahují vstupní vrstvu, která má stejný rozměr jako vstupní data. Následují skryté vrstvy, ve kterých počet skrytých vrstev a počet neuronů v jednotlivých vrstvách záleží na řešené úloze. Charakteristické pro tento typ architektury neuronových sítí je to, že neuron v jedné vrstvě je spojen s každým dalším neuronem ve vrstvě následující. [14]

Hlavní výhodou použití této architektury je možnost využití jak pro dopředné učení, tak pro algoritmus zpětného šíření chyby, kde se na základě ztrátové funkce nastavují parametry jednotlivých neuronů během učení neuronové sítě. [14]

Nevýhoda je ve velkém množství parametrů. To může být problém v případě, pokud se zpracovávají snímky. Například pro snímek, který má velikost 100x100 pixelů a má tři kanály (RGB), by bylo ve vstupní vrstvě potřeba mít 30 000 neuronů. Komplexnější úlohy jsou proto charakterizovány velkou výpočetní náročností. Tudíž je pro zpracování snímku výhodnější použít konvoluční neuronové sítě, které jsou popsány v další kapitole. [14]



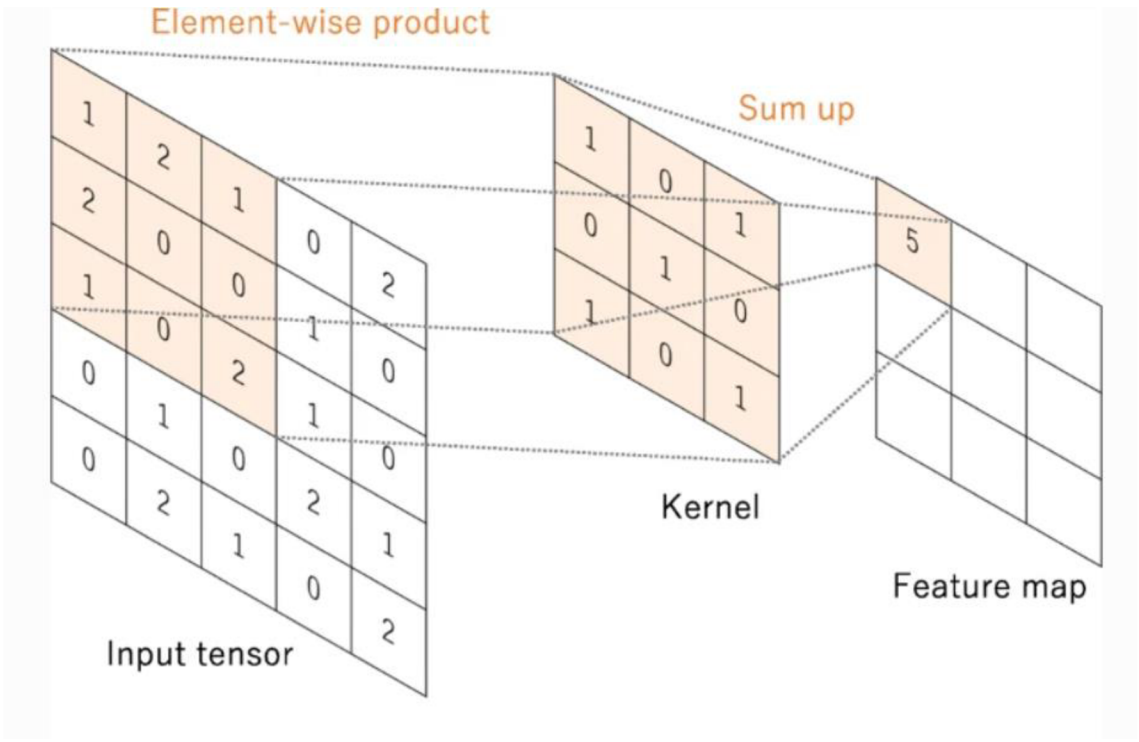
Obrázek 3.4 Architektura plně propojených vrstev [14]

3.3 Konvoluční vrstvy

Konvoluční neuronová síť se využívá v aplikacích, kde je velké množství vstupních dat. Je tedy výhodné ji využít, pokud pracujeme s dvourozměrnými daty (obrazová data). Pomocí konvolučních vrstev je možné model neuronové sítě naučit rozeznat vzory nižších i vyšších úrovní. [15]

3.3.1 Konvoluce

Konvoluce je matematická operace, která se využívá pro extrakci vzorů ze vstupního obrazu. Základem je matice o velikosti například 3x3, která se nazývá jádro. Tato matice je posouvána po vstupním obrazu, dále je prováděna matematická operace násobení po prvku mezi jádrovou maticí a submaticí vstupního obrazu. Nakonec je vypočtena suma jednotlivých násobení a výsledek je ukládán do výstupní matice na příslušnou pozici. Výstupem je snímek o stejné velikosti jako vstupní. Rozdíl je v tom, že ve výstupním snímku se nachází informace o vzorech, které se nachází ve vstupním snímku. Příklad konvoluce se nachází na obrázku 3.5. [15]



Obrázek 3.5 Příklad konvoluce [15]

Výhoda je v počtu parametrů. Počet parametrů odpovídá velikosti jádrové matice. V případě velikosti 3x3 a třech vstupních kanálů (RGB) bude počet parametrů jedné vrstvy, která obsahuje jednu jádrovou matici, roven 27. Konvolučních vrstev se většinou používá více za sebou, každá s jiným účelem. První vrstvy slouží jako

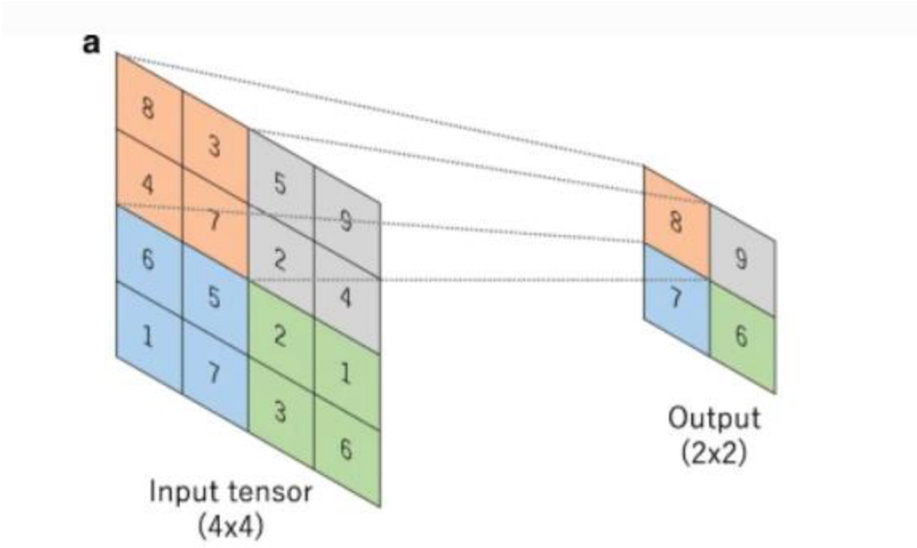
detektory různých hran, hlubší vrstvy slouží k propojení hran z prvních vrstev a k následnému rozpoznání složitějších objektů. [15]

Dalším parametrem konvoluční vrstvy je počet jádrových matic. Většinou se využívá více jádrových matic s odlišnou váhou každé matice. [15]

3.3.2 Pooling vrstva

Pomocí Pooling vrstev dojde ke zmenšení snímků, a tedy ke snížení výpočetní náročnosti. Hlavními typy Pooling vrstev jsou Max pooling a Average pooling. [15]

Max pooling vrstva funguje na principu toho, že je vybráno okno o rozměrech například 2x2. To je posouváno po vstupním obraze a zároveň je počítána maximální hodnota. Tato hodnota se nachází v tomto okně a následně se vybere jako výstup na příslušné pozici. Toto okno je ve snímku posouváno se stride 2, čímž dojde k redukci velikostí dimenzí vstupního snímku na polovinu. Princip je znázorněn na obrázku 3.6. [15]



Obrázek 3.6 Příklad principu Max pooling vrstvy [15]

Average pooling vrstva funguje podobným způsobem jako Max pooling. Rozdíl je ve výpočtu výstupní hodnoty, kde se místo maximální hodnoty vypočítává průměrná hodnota. [15]

3.3.3 Upsampling vrstva

Upsampling vrstva je další typ vrstvy, která slouží ke zvětšení vstupního snímku. Využívána je v autoenkodéru, když je potřeba z dělící vrstvy rekonstruovat snímek, který má stejnou velikost jako na vstupu. [16]

3.4 Ztrátová funkce

Ztrátová funkce se využívá k měření přesnosti predikcí, které jsou výstupem modelu neuronové sítě a skutečných predikcí. Využívá se různých typů ztrátové funkce, v závislosti na konkrétní aplikaci. Například v případě binárního klasifikátoru se využívá cross entropy. [17]

Další často používaná ztrátová funkce je Mean Square Error. Pro výpočet se využívá druhé mocniny rozdílu mezi predikovanými a skutečnými výstupy modelu, podělenými počtem zkoumaných prvků. Výhodné využití je v případech, kdy se zpracovávají obrazová vstupní data. Počítá se druhá mocnina rozdílu mezi skutečným a predikovaným snímkem pro každý pixel. Výsledek je poté podělen počtem pixelů ve snímku. [17]

3.5 Optimalizační algoritmus

Optimalizační algoritmy se využívají v neuronových sítích pro změnu parametrů sítě tak, aby došlo k redukce hodnoty ztrátové funkce. Změna parametrů sítě závisí na použitém optimalizačním algoritmu. Využívá se různých typů optimalizačních algoritmů, kde dva hlavní budou popsány dále. [18]

3.5.1 Gradient descent

Jedná se o nejjednodušší optimalizační algoritmus. Využívá se první derivace ztrátové funkce, čímž se získá gradient. Tím se získá směr, kterým by se měly parametry sítě měnit, aby bylo dosaženo minima ztrátové funkce. [18]

Výhody tohoto algoritmu jsou v jednoduchosti implementace. Nevýhody v možnosti uváznutí gradientu v lokálním minimu ztrátové funkce. Dále parametry sítě jsou aktualizovány až po vypočtení gradientu pro celý dataset. V případě velkých datasetu tedy může trvat dlouhou dobu, než dojde k nalezení minima ztrátové funkce. [18]

Kvůli poslední jmenované nevýhodě se využívá modifikovaná podoba Stochastic gradient descent. Tato modifikovaná verze algoritmu Gradient descent aktualizuje parametry sítě po každém zpracovaném vzorku datasetu oproti původnímu Gradient descent. Ten aktualizuje parametry sítě až po zpracování celého datasetu. Výhodou je tedy, že ke konvergenci dochází rychleji. [18]

3.5.2 ADAM

ADAM optimalizační algoritmus pracuje s momenty prvního i druhého řádu. To znamená, že zohledňuje nejen gradient samotný, ale i jeho derivaci. Learning rate, který vyjadřuje změnu parametrů mezi dvěmi iteracemi, se mění na základě vzdálenosti mezi minimem a aktuální pozici ztrátové funkce. Nastavuje se tedy maximální hodnota learning rate, která se v průběhu učení snižuje. [18]

Výhodou tohoto optimalizačního algoritmu je rychlost učení, nevýhodou výpočetní náročnost. [18]

3.6 Učení modelu neuronové sítě

Učení modelu neuronové sítě je proces, jehož cílem je nastavit parametry neuronové sítě tak, aby vznikl co nejmenší rozdíl mezi predikovaným a skutečným výstupem. [15]

3.6.1 Dataset

Získání správných dat vhodných pro využití je jednou z nejdůležitějších výzev pro vytvoření správného modelu neuronové sítě. Správnost výběru a zpracování dát má velký vliv na výsledky trénování sítě. Celý dataset se rozdělí na 3 hlavní částí, kterými jsou trénovací data, validační data a testovací data. Konkrétní poměr rozložení datasetu mezi tři složky může vypadat například 70 % trénovací data, 20 % validační data a 10 % testovací. [15]

Trénovací dataset je určen k trénování modelu neuronové sítě. To znamená, že pomocí tohoto datasetu se síť učí hledat v těchto datech vzory a informace. Této vlastnosti se říká generalizace. Vyhodnocuje se trénovací ztrátová funkce, na jejíž základě se aktualizují parametry neuronové sítě.[15]

Validační dataset slouží k vyhodnocení přesnosti modelu na datech, která síť nikdy předtím neviděla. To znamená, že pomocí validačních dat se zjistí, jak dobře je navržená a natrénovaná síť schopná generalizovat a najít vzory a informace v těchto datech. Vyhodnocuje se validační ztrátová funkce, na jejíž základě se ladí hyperparametry (learning rate, počet vrstev, počet neuronů ve vrstvách, počet epoch) sítě. [15]

Testovací data se využívají pro finální zhodnocení přesnosti modelu. [15]

3.6.2 Učení Backpropagation

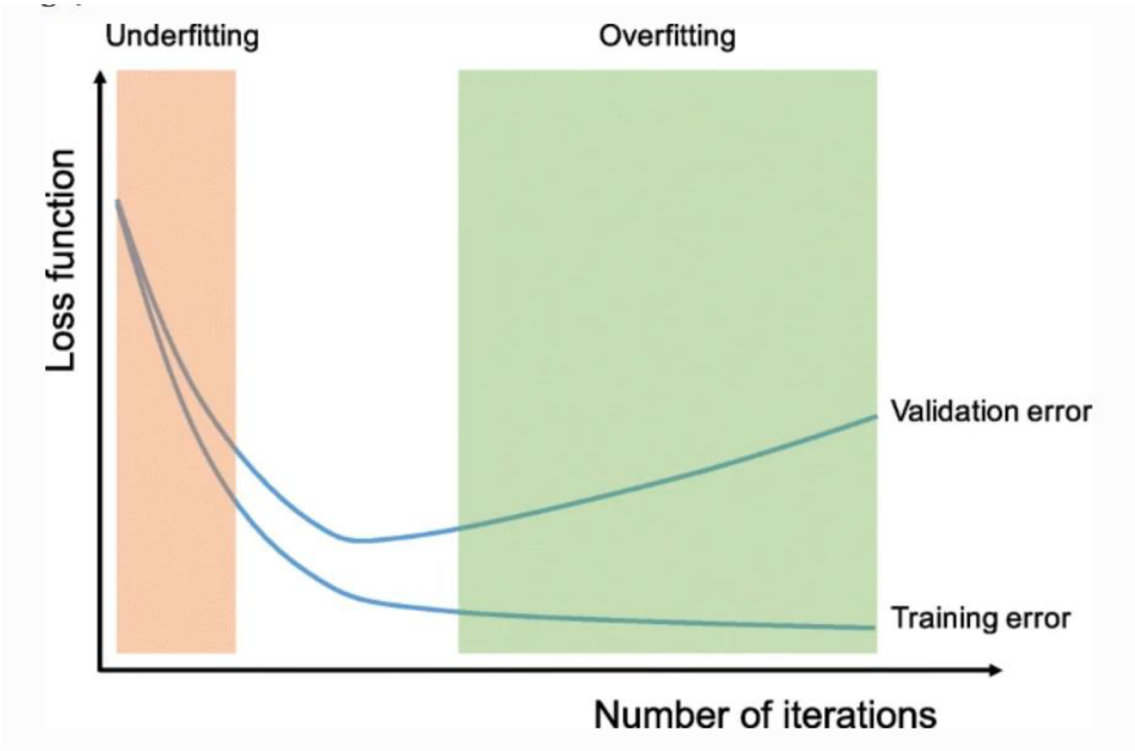
Cílem učení je nastavit parametry modelu tak, aby byl model co nejlépe přizpůsoben vstupním datům. Učení sítě pomocí metody Backpropagation zahrnuje dva hlavní kroky.

Prvním je dopředný chod. To znamená, že na vstupní vrstvu sítě se vloží data, postupně se vypočítávají výstupy jednotlivých neuronů ve skrytých vrstvách, a nakonec se vypočítají výstupní hodnoty pro výstupní vrstvu. [19]

V druhém kroku je chod opačný. To znamená, že při zpětném chodu se začíná na výstupní vrstvě a pokračuje přes skryté vrstvy až na vstupní vrstvu. Na základě ztrátové funkce dochází k výpočtu, jakým způsobem musí jednotlivé neurony změnit své parametry tak, aby došlo k snížení ztrátové funkce. [19]

3.6.3 Testovací a trénovací ztrátová funkce

V průběhu učení neuronové sítě se počítají dvě ztrátové funkce. Jedna je pro trénovací data, druhá pro validační data. Na základě těchto hodnot v průběhu učení modelu neuronové sítě lze určit, jakým směrem se učení vyvíjí. Možnosti, ke kterým může dojít, jsou zobrazeny na obrázku 3.7. [15]



Obrázek 3.7 Průběhy trénovací a validační ztrátové funkce [15]

3.6.4 Overfitting

Overfitting je stav, kdy po poklesu validační chyby dojde k následnému růstu. Je to vidět na obrázku 3.7 v zelené části. Zatímco trénovací chyba se pomalu zmenšuje, validační chyba roste. Důvodem je to, že dochází k přeučování neuronové sítě. To znamená, že neuronová síť se začíná učit a přizpůsobovat pouze trénovacím datům, ale

není schopna generalizace. Pokud se poté na vstupu použijí data, která nikdy neviděla, není schopna je dostatečně přesně zpracovat. [15]

Metody, jak předejít přeučení neuronové sítě, jsou různé. Nejlepším řešením je získat více dat. Model, který je trénován na větším množství dat, bude lépe generalizovat. Dalšími možnostmi mohou být data augumetation, dropout nebo snížení komplexnosti neuronové sítě. [15]

3.6.5 Underfitting

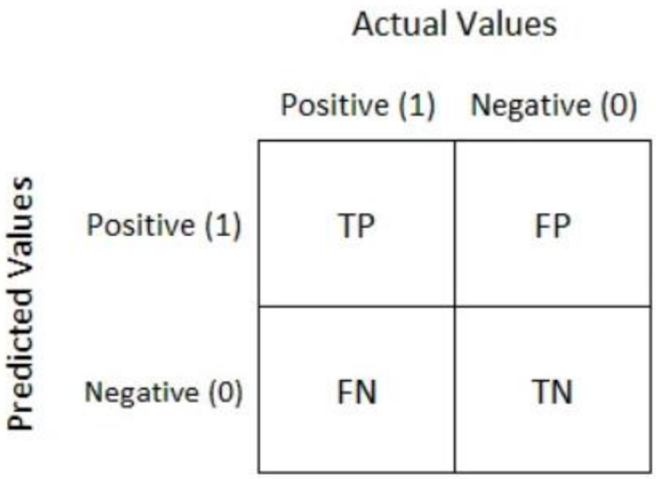
Underfitting je stav, kdy je trénovací chyba níže než validační, ale zároveň obě jsou na vysoké úrovni. To znamená, že model není schopen dostatečně se naučit na vstupních datech, což generuje velkou chybu na trénovacích i validačních datech. Hlavní možností, jak předejít underfittingu, je trénování modelu po delší dobu. [15]

3.7 Vyhodnocení modelu

Po celém procesu od získání vhodných dat, navržení architektury neuronové sítě a naučení modelu, je potřeba zhodnotit finální přesnost navrhnutého modelu. K tomu se využívají různé metriky, následně bude popsána matice záměn. [20]

3.7.1 Matice záměn

Jedná se o matici, kde výstupem jsou 4 různé kombinace mezi predikovaným a skutečným výstupem. Tabulka se nachází na obrázku 3.8. [20]



Obrázek 3.8 Matice záměn [20]

Je vidět, že obsahuje 4 hodnoty:

TP je zkratka pro True Positive. Jedná se o pozitivní predikce modelu, které jsou správné.

FP je zkratka pro False Negative. Vyjadřuje pozitivní predikce modelu, které jsou chybné.

FN je zkratka pro False Negative. Vyjadřuje negativní predikce modelu, které jsou chybné.

TN je zkratka pro True Negative. Vyjadřuje negativní predikce modelu, které jsou správné. [20]

Na základě těchto hodnot lze vypočítat další metriky, jako je celková správnost, senzitivita a specificita. Tyto metriky jsou popsány v rovnicích (3.1), (3.2) a (3.3) Čím vyšší jsou tyto hodnoty, tím přesnější a spolehlivější je navržený model. [20]

$$celková\ správnost = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.1)\ [20]$$

$$senzitivita = \frac{TP}{TP + FN} \quad (3.2)\ [20]$$

$$specificita = \frac{TN}{TN + FP} \quad (3.3)\ [20]$$

3.8 PyTorch

PyTorch je optimalizovaná knihovna zaměřena pro aplikaci hlubokého učení neuronových sítích, která je založena na programovacích jazycích Python a Torch. [21]

Základem knihovny PyTorch jsou tensory, které si lze představit jako kontejnery pro uchovávání dat. Data mohou mít více dimenzí. Příkladem využití je práce s datasetem, který obsahuje snímky. Takový tensor bude mít čtyři dimenze. Rozměry pak budou (*počet snímků, počet kanálů, výška, šířka*). [21]

3.8.1 Moduly knihovny PyTorch

PyTorch obsahuje několik modulů, které se využívají při návrhu modelu neuronové sítě. [21]

Hlavním je nn modul, který obsahuje různé třídy pro vytvoření architektury neuronové sítě. Příkladem jsou konvoluční vrstvy nebo plně propojené vrstvy. [21]

Další je modul Optim, který obsahuje algoritmy pro optimalizaci. Například tedy obsahuje algoritmy Adam nebo SGD. [21]

3.9 Google Colab

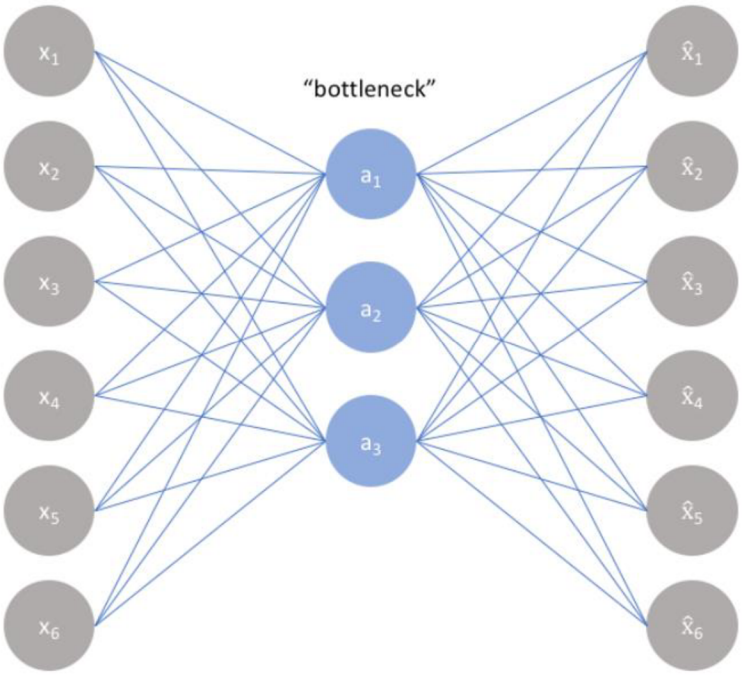
Jedná se o produkt společnosti Google, který umožňuje psát a provádět kód psaný v jazyce Python pomocí webového prohlížeče a využívající výpočetních zdrojů jako je CPU a GPU. Jedná se o cloudové řešení, to znamená, že není potřeba nic stahovat ani instalovat. Jako uložiště se využívá Google Drive. [22]

Výhodné je využití pro úlohy strojového učení. Využiti v základní verzi je bezplatné, avšak oproti placené verzi je omezené. K dispozici je menší paměť RAM a časově omezená dostupnost výpočetních zdrojů GPU. Pro jednodušší aplikace stačí free verze, pro složitější úlohy by nemusela být dostačující. [22]

4. AUTOENKODÉR

Autoenkodér je architektura neuronové sítě. Jejím účelem je redukovat vstupní data do prostoru, který má menší velikost (extrakce nejvýznamnějších vzoru a informací) a z tohoto redukovaného prostoru rekonstruovat vstupní data nazpět s co nejmenší rekonstrukční chybou. Jedná se tedy o učení bez učitele, jelikož nejsou předem známy výstupní hodnoty. Ty se generují na základě vstupních data, nastavení parametrů a architektury sítě. [23]

Obecná a jednoduchá architektura autoenkodéru je na obrázku 4.1. Je vidět, že vstupní vrstva a výstupní vrstva mají stejný rozměr. Dále mezi nimi se nachází dělící vrstva, která má rozměr menší. Do této dělící vrstvy se redukují vstupní data. Z této vrstvy se následně rekonstruují zpět vstupní data s co nejmenší odchylkou. [23]

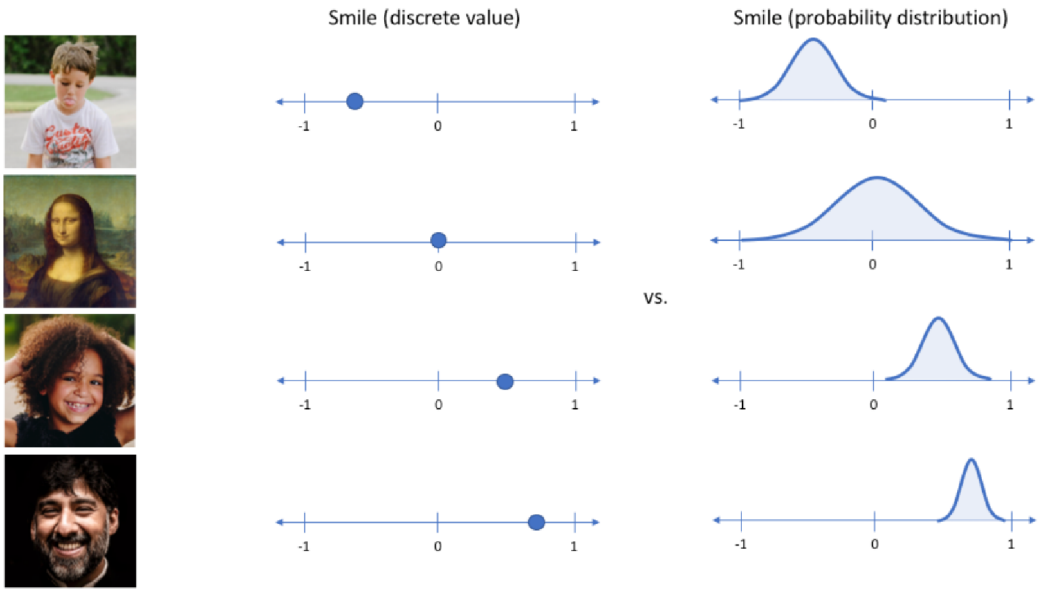


Obrázek 4.1 Obecná architektura autoenkodéru [23]

Autoenkodéry lze dále dělit na pět typů. Undercomplete, Sparse, Contractive, Denoising a Variational. Poslední jmenovaný je využit v této práci, proto dále bude popsána architektura variačního autoenkodéru. [24]

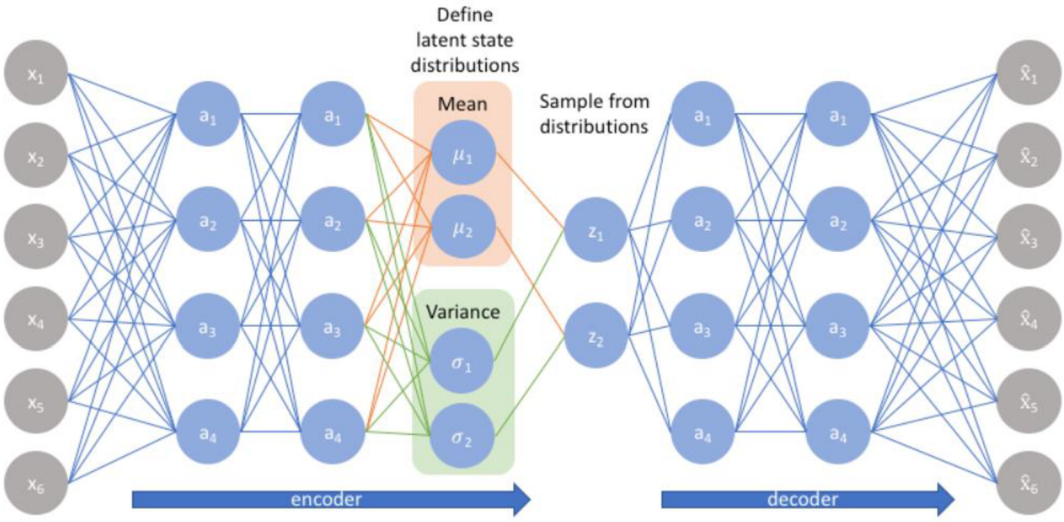
Variační autoenkodér se liší od běžného Autoenkodéru v uspořádání dělící vrstvy. Zatímco klasický Autoenkodér zde má pouze jeden vektor reprezentující latentní prostor, variační Autoenkodér obsahuje dva vektory, které reprezentují latentní prostor. První z nich vyjadřuje průměr, druhý z nich rozptyl. Znamená to tedy, že latentní

prostor pro každý vstup je vyjádřen pomocí pravděpodobnostního rozdělení, zatímco pro klasický Autoenkodér je každý atribut latentního prostoru vyjádřen pomocí čísla. Znázornění se nachází na obrázku 4.2. Při rekonstrukci se z jednotlivých atributů latentního prostoru použije náhodná hodnota z rozložení, aby se vygeneroval vektor pro následnou rekonstrukci. [24]



Obrázek 4.2 Porovnání latentního prostoru pro klasický a variační autoenkodér [24]

Autoenkodér se dělí na tři části – enkodér, dekodér a dělící vrstva. Obecná architektura je na obrázku 4.3.



Obrázek 4.3 Architektura variačního autoenkodéru [24]

4.1 Enkodér

Jedná se o první část variačního autoenkodéru. První vrstva enkodéru má v případě využití plně propojených vrstev stejný rozměr (počet neuronů) jako vstupní data. S každou další vrstvou dojde k postupnému snižování počtu neuronů v jednotlivých vrstvách. Úkolem enkodéru je tedy redukovat dimenzi dat způsobem, který zachytí nejvýznamnější vzory a informace z původních dat. [24]

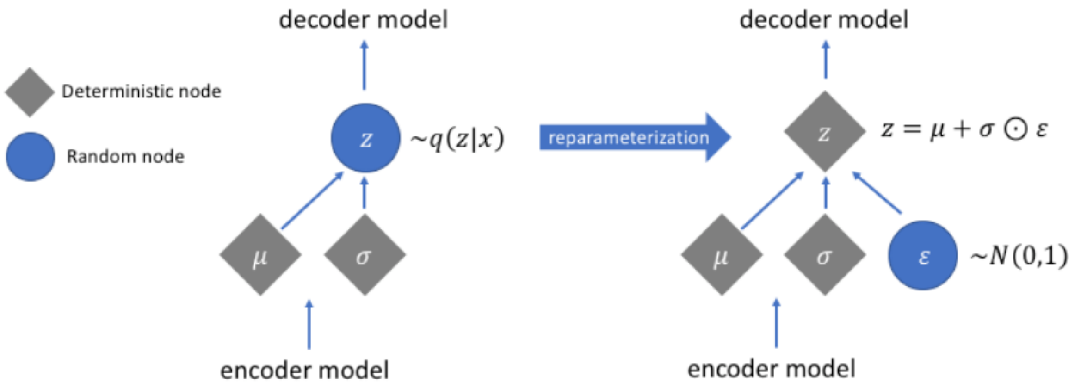
V případě obrazových dat je však výhodnější využít konvolučních bloků, které mají mnohem méně parametrů, zároveň však dokáží lépe extrahovat ze vstupního snímku vzory a informace. [24]

4.2 Dělící vrstva

Jedná se o vrstvu, která má nejmenší dimenzi. Vstupní data se redukují do velikosti dělící vrstvy, která obsahuje dva vektory. Jelikož však při rekonstrukci dochází k náhodnému výběru hodnot z rozložení jednotlivých atributů, nemohlo by být následně využito při trénováni metody Backpropagation. [24]

4.2.1 Reparametrizační trik

Z důvodu nemožnosti využít Backpropagation se používá tzv. reparametrizační trik, což je znázorněno na obrázku 4.4. V případě využití reparametrizace vektor z není nadále náhodného charakteru. Tím lze využít Backpropagation při učení této sítě. [24]



Obrázek 4.4 Reparametrizační trik [24]

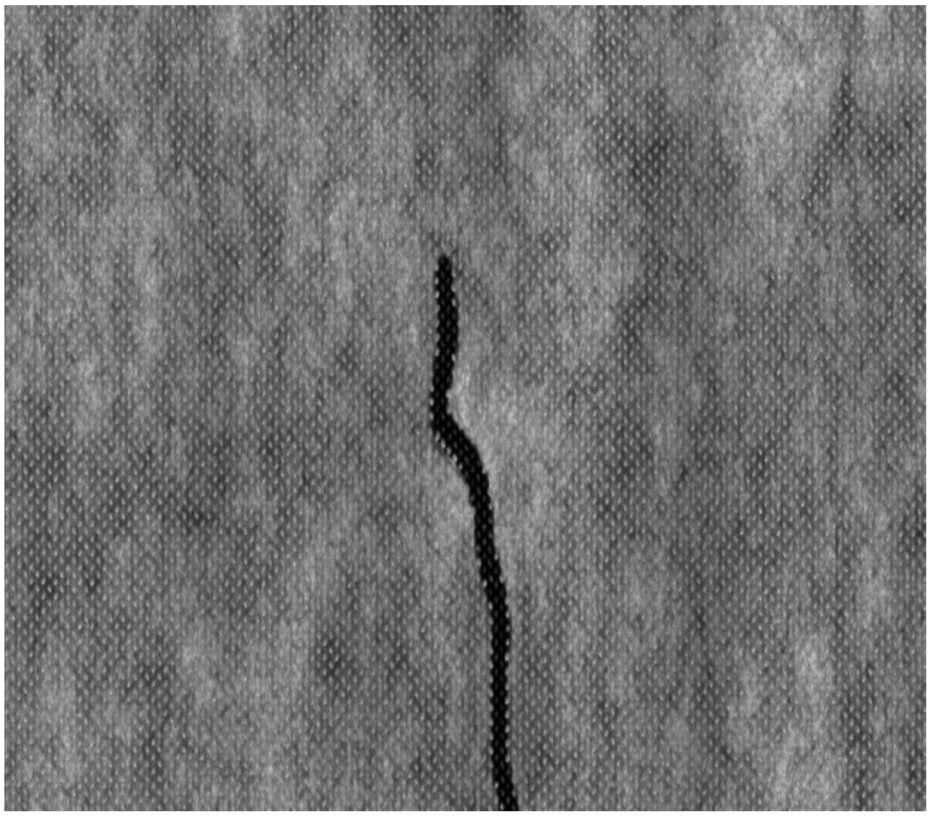
4.3 Dekodér

Jedná se o poslední část variačního autoenkodéru. Vstupem je vektor z, ze kterého se pomocí dalších vrstev rekonstruuje, v ideálním případě s co nejmenší chybou, původní snímek. Poslední vrstva dekodéru má stejný rozměr jako první vrstva enkodéru. Využívá se Upsampling vrstev pro zvýšení dimenze dat. [24]

5. DETEKCE ANOMÁLIÍ VE VZORU

Proces kalandrování byl popsán v kapitole 1. Jako v každém procesu, mohou se i při kalandrování vyskytnout vady, které ovlivní kvalitu výsledného výrobku. V případě netkaných textilií, které byly upraveny procesem kalandrování, se může jednat o vadu v netkané textilii nebo vadu otisku gravury pomocí kalandrování. Byly dodány snímky, na kterých jsou tyto vady zřetelné.

Příklad vady netkané textilie je znázorněn na obrázku 5.1. Je vidět, že pravděpodobně došlo ke kontaminaci vrstvy netkané textilie tlustým černým vláknem.

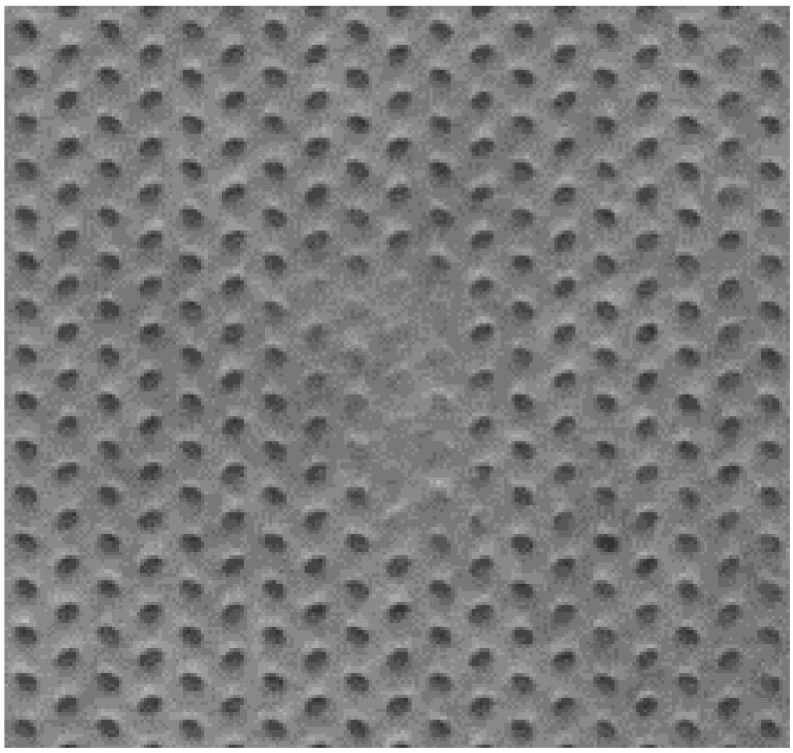


Obrázek 5.1 Vada kontaminace černým vláknem na netkané textilií

Detekce těchto vad však není předmětem této diplomové práce, proto se dále zaměříme na vady, které vznikají při procesu kalandrování.

Příklad vady, která vznikla při procesu kalandrování, je na obrázku 5.2. Je vidět, že se tato vada nenachází na vrstvě netkané textilie, ale jedná se o vadu gravury, která vznikla při kalandrování. Tato vada může vzniknout v případě opotřebování kalandrovacího válce. Proto se tato vada bude cyklicky opakovat každou otočku kalandrovacího válce. Této vlastnosti by tedy mohlo být využito v případě detekce, kdy by se určitá oblast zvolila jako anomální pouze v případě, pokud by tato oblast byla detekována vícekrát. Je taktéž vidět, že se jedná pouze o vady gravury, které nijak nezasahují do jeho okolí. Pro detekci budou otestovány metody, které detekují pozici

pouze jednoho chybějícího kalandru. Tyto algoritmy by bylo možno lehce upravit tak, aby detekovaly všechny chybějící kalandry.



Obrázek 5.2 Vada vzniklá při procesu kalandrování

Otestovány byly tři metody. První metoda je založena na převodu snímku do binární podoby pomocí Fourierovy transformace, kde gravura je vyjádřena bílou barvou, netkaná textilie černou. Zde se pomocí výřezu, který je posouván po snímku, vyhodnocuje, jestli se v tom aktuálním okně nachází dostatečný počet bílých pixelů, které vyjadřují gravuru. V místech, kde se počet bílých pixelů bude blížit nule, je pravděpodobně anomálie-chybějící gravura.

Druhá otestovaná metoda využívala pouze Fourierovy transformace. Pracovalo se s předpokladem, že rozdíl snímku netkané textilie ve frekvenční oblasti, na které není anomálie, a snímku textilie ve frekvenční oblasti, na které je anomálie, vytvoří nový snímek. Tím bude zvýrazněno místo, kde je chybějící gravura.

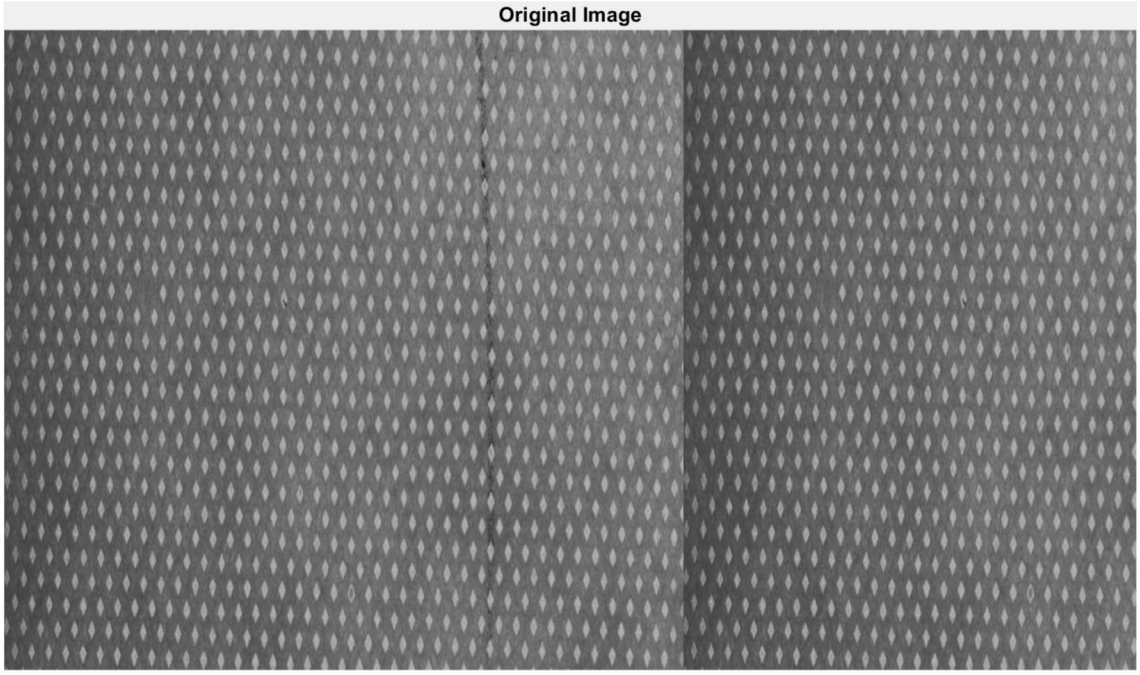
Poslední otestovanou metodou bylo využití neuronových sítí. Zde byla využita knihovna PyTorch, pomocí které byl navrhnut model variačního Autoenkodéru. Tato metoda se ukázala být jako účinná a vhodná vzhledem k povaze úlohy. Dodány byly snímky, které byly předzpracovány a následně využity jako dataset pro naučení modelu. Tím se model naučil vzor gravury. Pokud byl na vstupu použit snímek, kde gravura chyběla, síť dokázala tuto gravuru na příslušné místo doplnit. Následně se vypočítal rozdíl vstupního a výstupního snímku, kde bylo zvýrazněno místo s anomálií. Nakonec se toto místo nalezlo pomocí knihovny OpenCV a funkce MatchTemplate nebo

vyhledání všech Contours a vyhodnocení rozměrů na základě rozptylu hodnot x a y. Jednotlivé postupy budou podrobněji rozebrány dále.

5.1 Statistická detekce anomálií

Pro detekci anomálie v dodaném vzorku netkané textilie s gravurou bylo využito právě Fourierovy transformace a její možností. Pro analýzu a následné řešení tohoto zadání byla využita platforma Matlab, která je vhodná pro zpracování snímků a manipulaci s nimi. Dále byly dodány příklady netkané textilie s gravurou s chybějícím kalandrem, který tedy bylo potřeba detekovat. Typů snímku s různými gravurami bylo dodáno několik. Následně je popsáno možné řešení úlohy pro jednu z nich. Snímek s netkanou textilií a kosočtvercovou gravurou se nachází na obrázku 5.3.

5.1.1 Originální snímek



Obrázek 5.3 Snímek netkané textilie s kosočtvercovou gravurou

Jedná se o dodaný snímek, je vidět, že je vlastně vytvořen složením z jednoho snímku. Na snímku jsou dva chybějící kalandry, které je třeba detekovat.

Tento snímek byl načten do workspace Matlabu pomocí příkazu

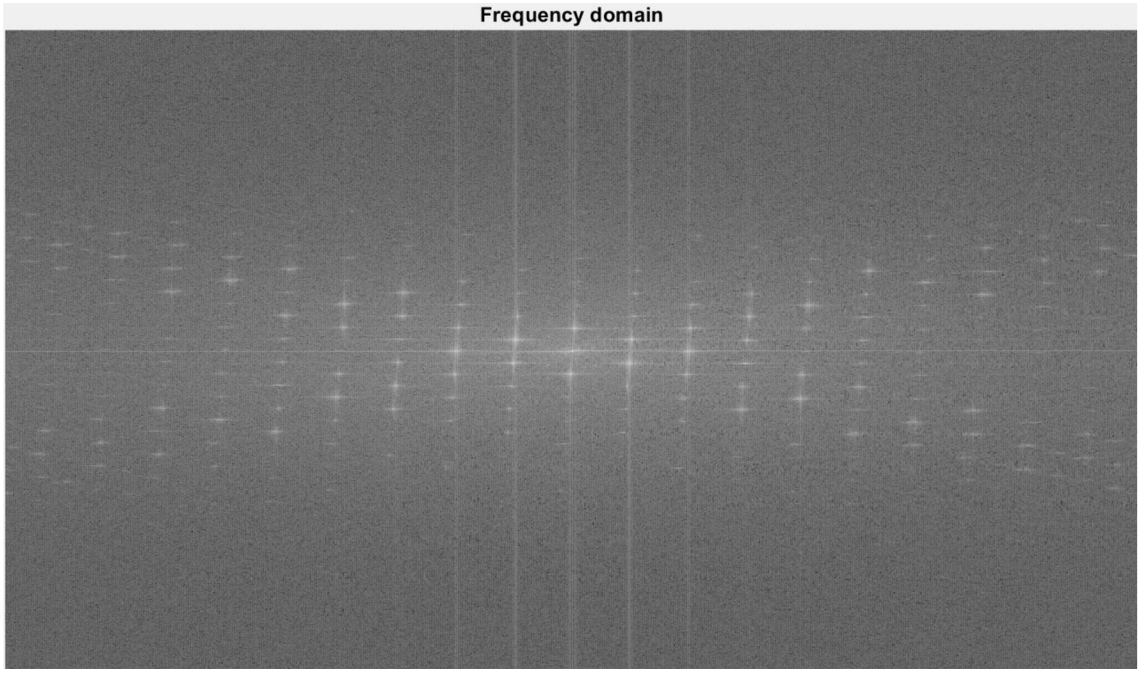
```
grayImage = imread("Kosoctverce.jpg");
```

5.1.2 Frekvenční spektrum

Originální obrázek je následně převeden do frekvenčního spektra. To zajišťuje část kódu níže, kde zároveň dojde k posunu stejnosměrné složky do středu. Dále je výsledkem operace Fourierovy transformace komplexní číslo. Následně je pro výpočty použita pouze amplituda.

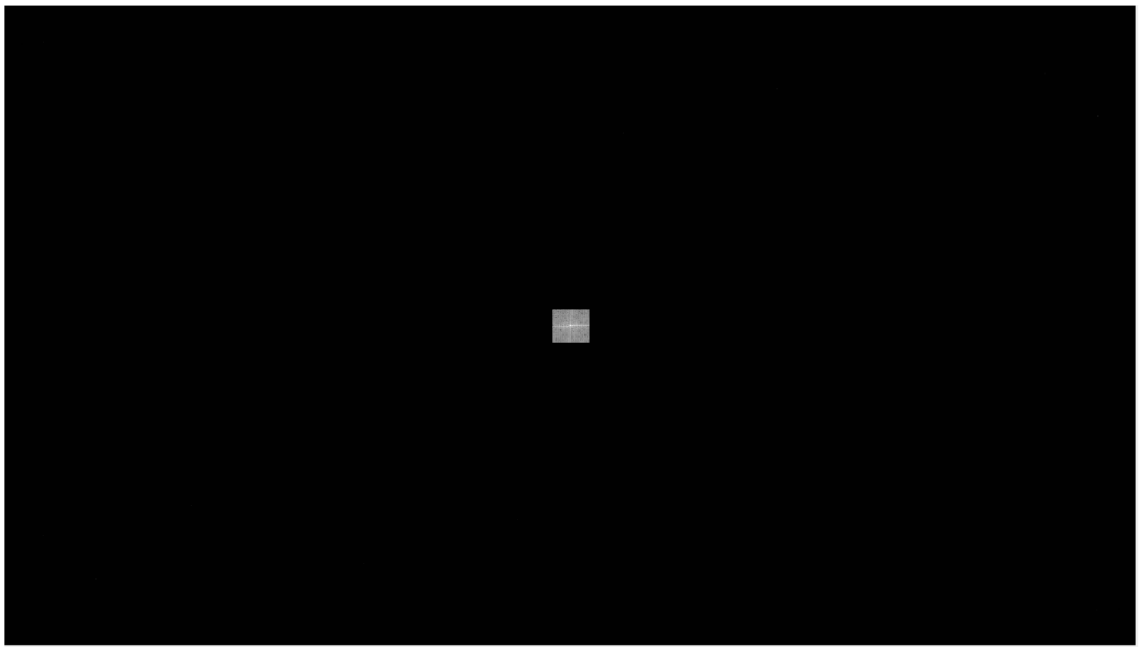
```
frequencyImage = fftshift(fft2(grayImage));
amplitudeImage = log(abs(frequencyImage));
```

Amplitudová část frekvenčního spektra původního obrázku je vidět na obr. 5.4.



Obrázek 5.4 Frekvenční spektrum snímků

Z něj je vidět, že na snímku se vyskytuje velké množství frekvencí, které tvoří popis snímku ve frekvenční oblasti. Pro další zpracování je vybrána pouze oblast s nejnižší frekvencí (jedná se o filtr typu dolní propusť), čímž se z původního snímku vytáhne pozadí, bez gravury. Frekvenční spektrum po filtraci je zobrazeno na obrázku 5.5. Je vidět, že je využit filtr s ostrými přechody, který je pro tuto aplikaci dostačující.

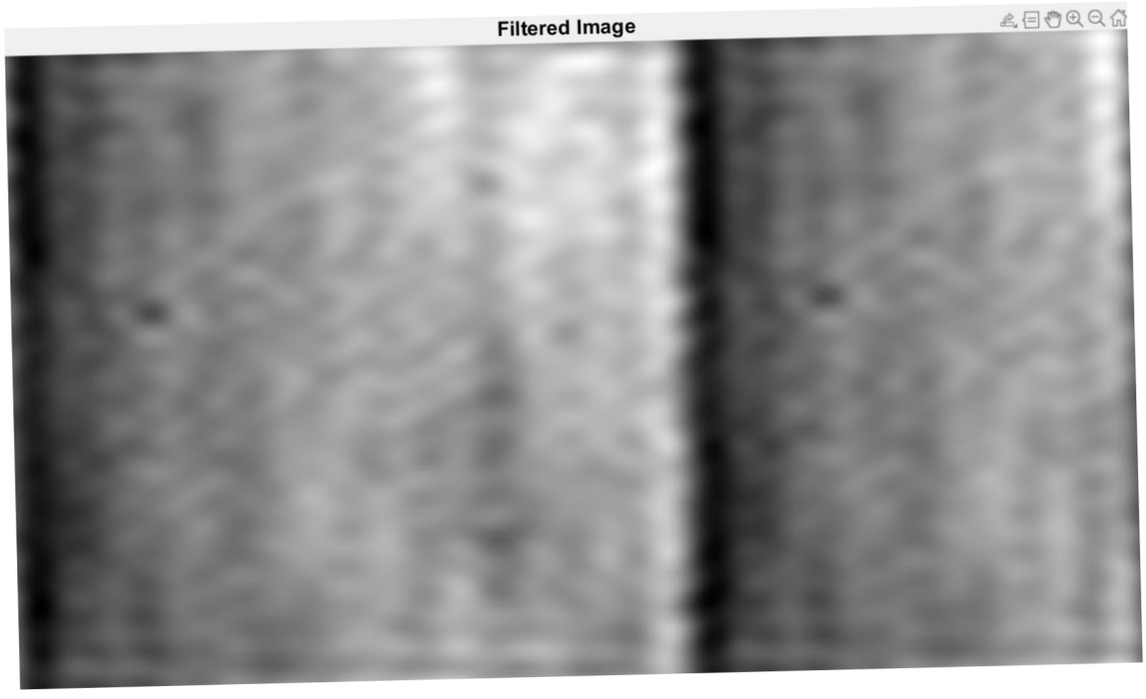


Obrázek 5.5 Frekvenční spektrum po filtraci

5.1.3 Převod zpět do prostorové oblasti

Následně je filtrované spektrum převedeno zpět do prostorové oblasti pomocí příkazů níže. Tím byl získán snímek, který obsahoval pouze pozadí původního snímku. Lze to vidět na obrázku 5.6.

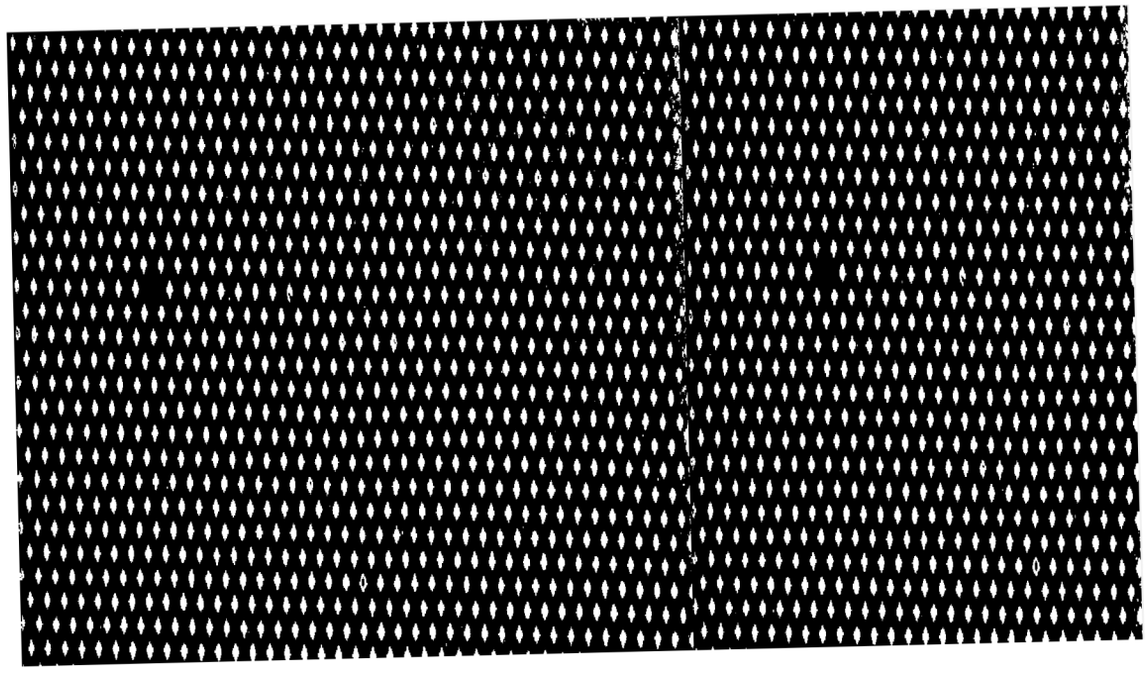
```
filteredImage = ifft2(fftshift(frequencyImage));
amplitudeImage3 = abs(filteredImage);
```



Obrázek 5.6 Filtrovaný obraz

Dále byl odečten původní snímek od snímku pouze s pozadím Tím byl získán snímek, na kterém byla patrná pouze gravura. Bylo využito prahování, aby byl získán pouze binární snímek pro další zpracování. Lze to vidět na obrázku 5.7.

```
b = grayImage- uint8(amplitudeImage3);
    a = b>15;
```



Obrázek 5.7 Gravura v binární podobě

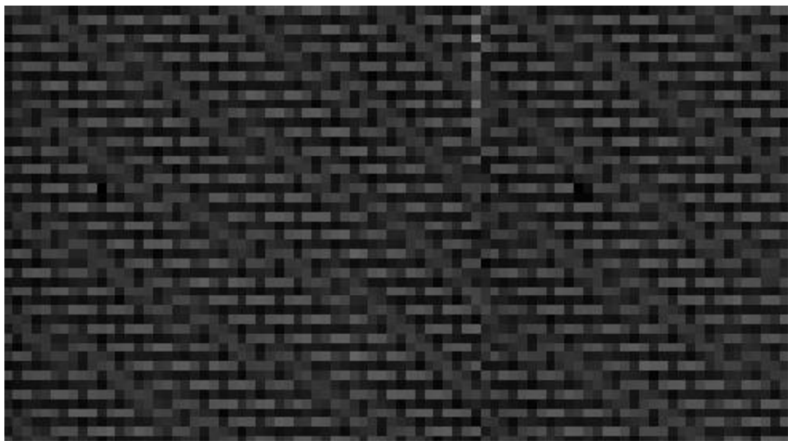
5.1.4 Detekce anomálii

Pro detekci anomálie ve vzoru se využívá součtu bílých v pixelů v určitém výřezu snímku. Pracuje se s předpokladem, že pokud bude zvolen výřez snímku s vhodnou velikostí, potom počet bílých pixelů v oblasti, kde chybí kalandr, se bude blížit nule (v ideálním případě, v praxi se však může vyskytnout ve snímku šum).

V této konkrétní aplikaci byl využit výřez snímku o velikosti 30x30 pro výpočet sumy bílých pixelů v této oblasti. Dále tento výřez byl posouván s každou iterací o polovinu své hodnoty velikosti. Následně bude půlka plochy výřezu využita pro výpočet v každé další iteraci. Výsledná hodnota z každé oblasti byla poté uložena do jiného dvourozměrného pole, čímž vlastně vznikl snímek, jehož jas vyjadřuje počet bílých pixelů v prohledávané oblasti. Lze ho vidět na obrázku 5.8.

Kód zajišťující tuto funkci je níže. Proměnná `size_element` vyjadřuje velikost výřezu. V tomto případě je nastavena na hodnotu 30 pixelů. Proměnná `sub` vyjadřuje překrytí. V tomto případě bude překrytí mezi následujícími snímky poloviční, což je vyjádřeno hodnotou 2.

```
for i = 0:x1
    for j=0:y1
    suma      =      a((i*size_element/sub)+1:((i+1)*size_element/sub)+1
,(j*size_element/sub)+1:((j+1)*size_element/sub)+1);
    numWhitePixels(i+1,j+1) = sum(suma(:));
    end
end
```



Obrázek 5.8 Snímek vyjadřující počet bílých pixelů v určitém výřezu snímku

Na tomto snímku jsou pouze dvě oblasti, kde se hodnota blíží k nule (v tomto případě se zde nevyskytuje šum, takže hodnota je rovna 0). Následně pomocí jednoduchého prahování je vytvořen binární snímek, který je na obrázku 5.9. Zde je jednoznačně,

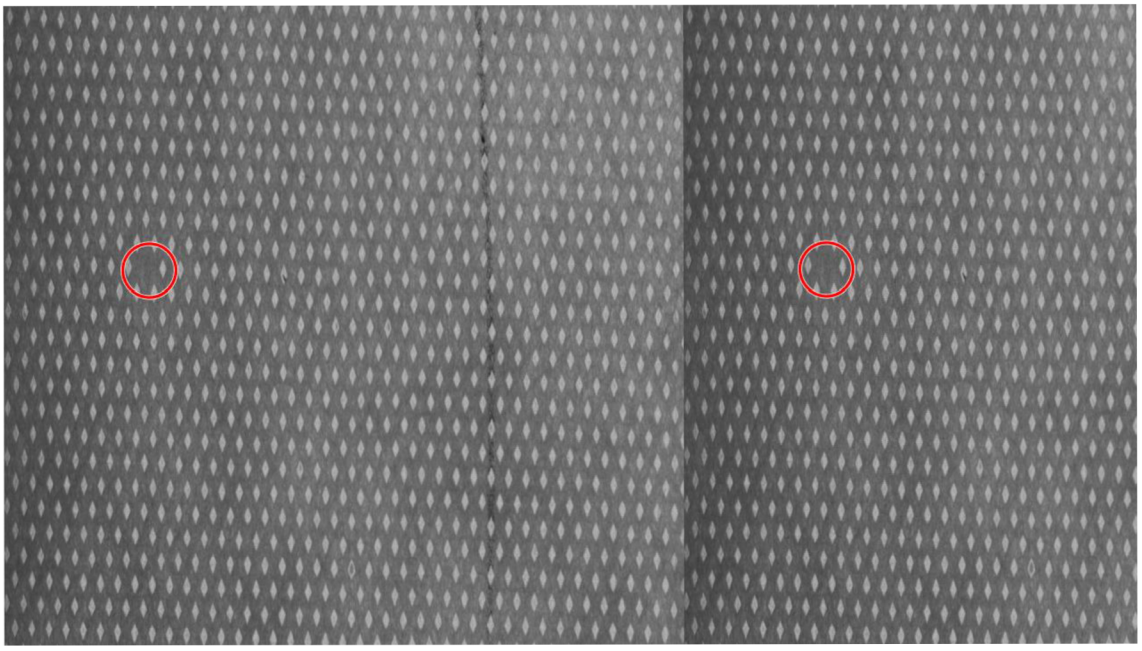
vidět, na kterých pozicích nebyly nalezený žádné bílé pixely, tedy pozice s chybějící gravurou. Tyto pozice jsou nalezeny pomocí kódu, který je níže.

```
regs = regionprops(h, 'Centroid');
```

Obrázek 5.9 Prahovaný snímek

5.1.5 Vyhodnocení

Tyto hodnoty jsou následně převedeny zpět na souřadnice v původním snímku. Převod proběhl na základě nalezené pozice bodů ve snímku 5.9, velikosti výřezu a velikosti posunu výřezu mezi iteracemi. Nakonec jsou tyto pozice vyznačeny kruhem pro ověření správnosti vyhodnocení algoritmu. To je vidět na snímku 5.10.

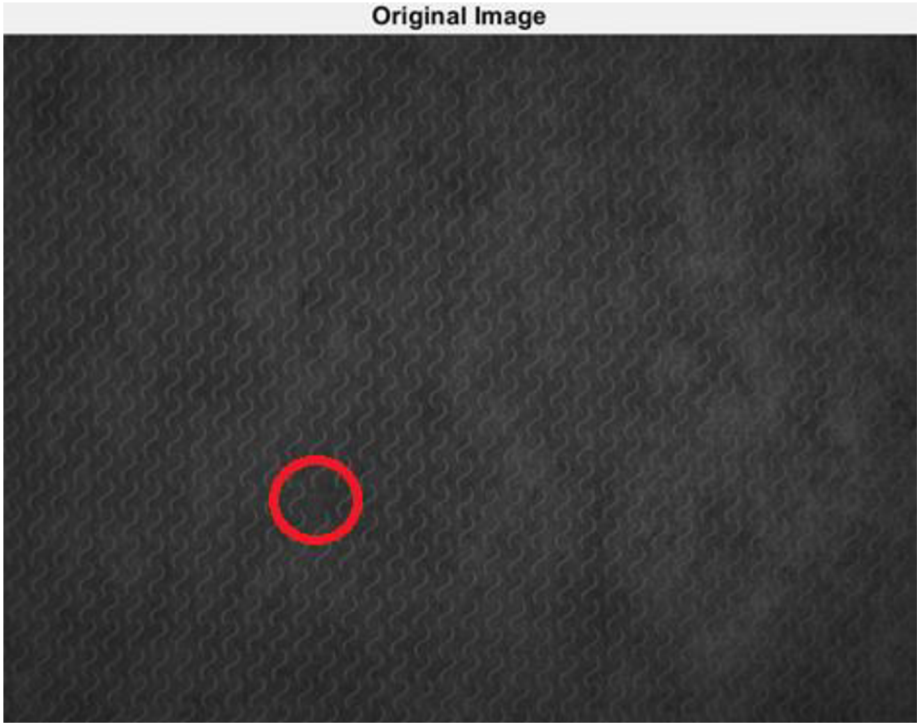


Obrázek 5.10 Výsledný snímek s detekovanými chybějícími kalandry

5.2 Detekce anomálií pomocí Fourierovy transformace

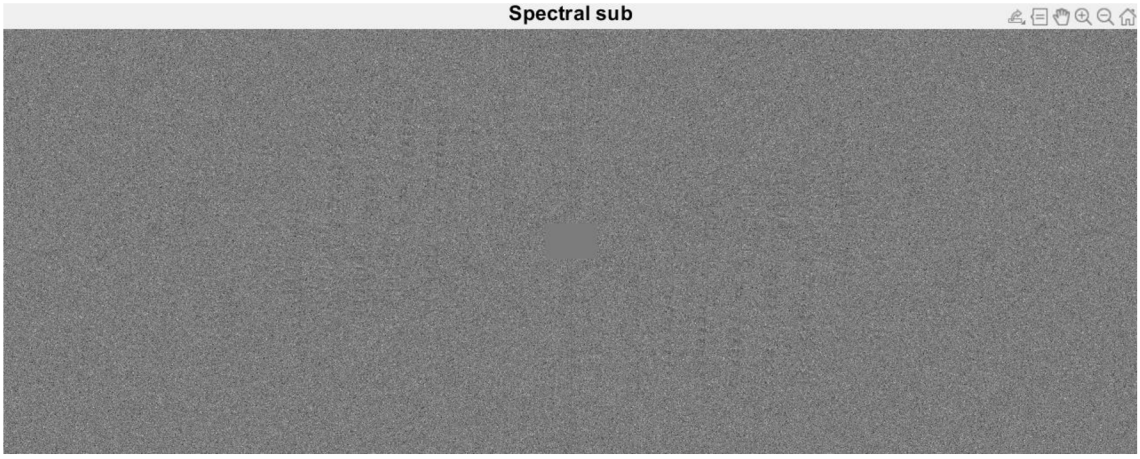
Druhou implementovanou metodou bylo využití Fourierovy transformace. Pracovalo se s předpokladem, že pokud se porovná frekvenční spektrum netkané textilie, kde je anomálie, se spektrem, kde se anomálie nenachází, dojde k zvýraznění oblasti, kde se anomálie nacházela.

Analyzovaný snímek je na obrázku 5.11. V tomto případě byla použita vlnková gravura. Je vidět, že se v něm nachází jedna chybějící gravura, která je zvýrazněna červeným kruhem.



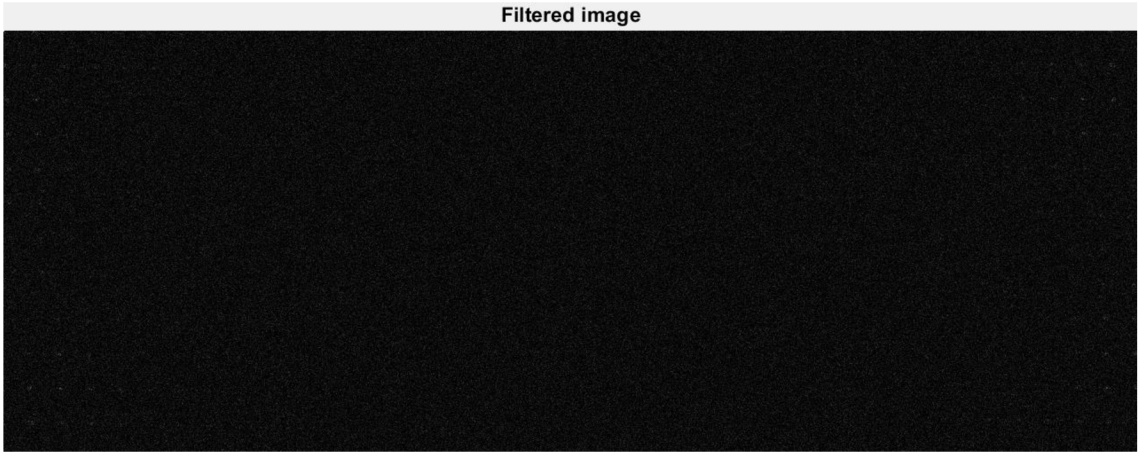
Obrázek 5.11 Analyzovaný snímek

Tento snímek byl zpracován podobným způsobem jako v předcházející úloze. S rozdílem, že se tento snímek rozdělil na dvě části – část s anomálií a část bez anomálie. Následně došlo k převedení obou snímků do frekvenční oblasti, kde se tentokrát využil filtr pro potlačení nízkých frekvencí. Dále se vypočítal rozdíl těchto dvou frekvenčních spekter. Výsledek je na obrázku 5.12.



Obrázek 5.12 Rozdíl ve frekvenční podobě

Po převodu rozdílu zpět do prostorové oblasti se získal tento výsledný snímek, na kterém není zřetelné nic z původního snímku. To je vidět na obrázku 5.13.



Obrázek 5.13 Výsledný snímek v prostorové oblasti

Byl vyzkoušen další postup, kdy z původních dvou snímků byl pomocí mediánového filtru odstraněn šum, avšak kýženého výsledku dosaženo nebylo. Proto další výzkum metody pouze pomocí Fourierovy transformace dále nepokračoval.

5.3 Detekce anomálií pomocí neuronových sítí

Poslední implementovanou a otestovanou metodou byla detekce anomálií pomocí neuronových sítí. Bylo využito vlastnosti variačního autoenkodéru, který vyniká v rekonstrukci snímků. Dále je popsán postup implementace.

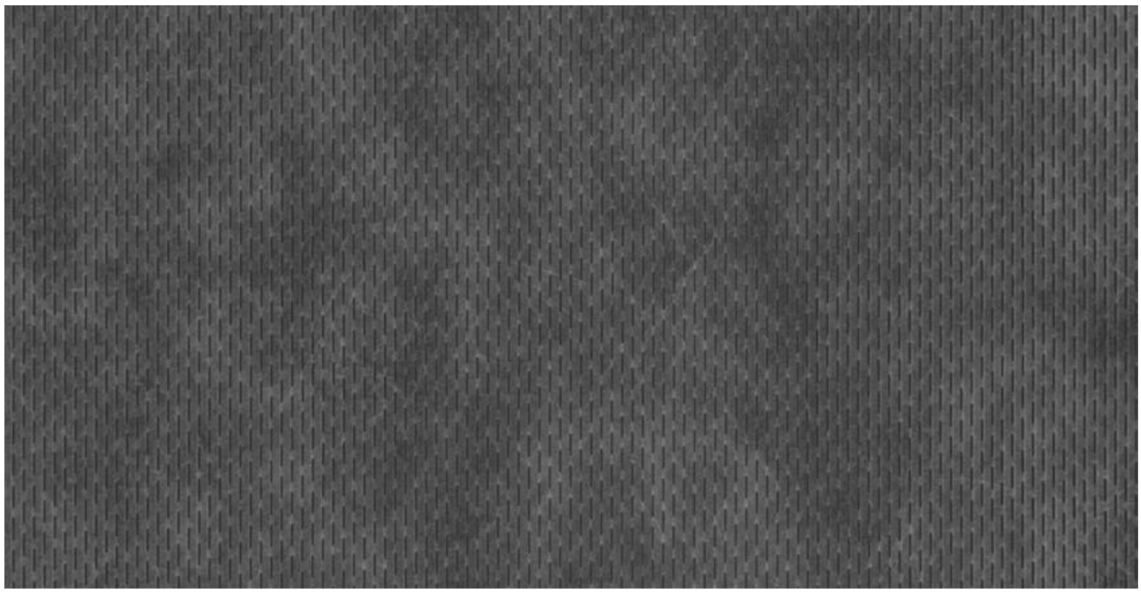
5.3.1 Data

Jednou z nejdůležitějších součástí vytvoření a natrénování správného modelu je výběr a následná úprava vhodných dat. Pro vypracování byl dodán dataset, který vznikl snímáním netkané textilie s proužkovou gravurou. Dále je rozdělen do dvou typů dle velikosti snímků. První typ je ve složce s názvem K1N, obsahuje 51 snímků a jeho rozlišení je 1000 x 6105 pixelů. Druhý typ je ve složce s názvem K2N a obsahuje 50 snímků, které mají rozlišení 1000 x 7230 pixelů. Z těchto dat byly náhodně vybrány celkově 4 snímky, které byly použity pro testovací účely. Tato data síť nikdy předtím neviděla. Nachází se ve složce Testovací data. Zbytek dat byl použit jako trénovací a validační data. Příklad výřezu snímku, který byl dodán a nejsou na něm žádné úpravy, je na obrázku 5.14. Takto malý dataset se může zdát jako nedostatečný, ale je třeba si uvědomit, že jako dataset slouží vlastně celé snímky s tisíci vzory v každém snímku.

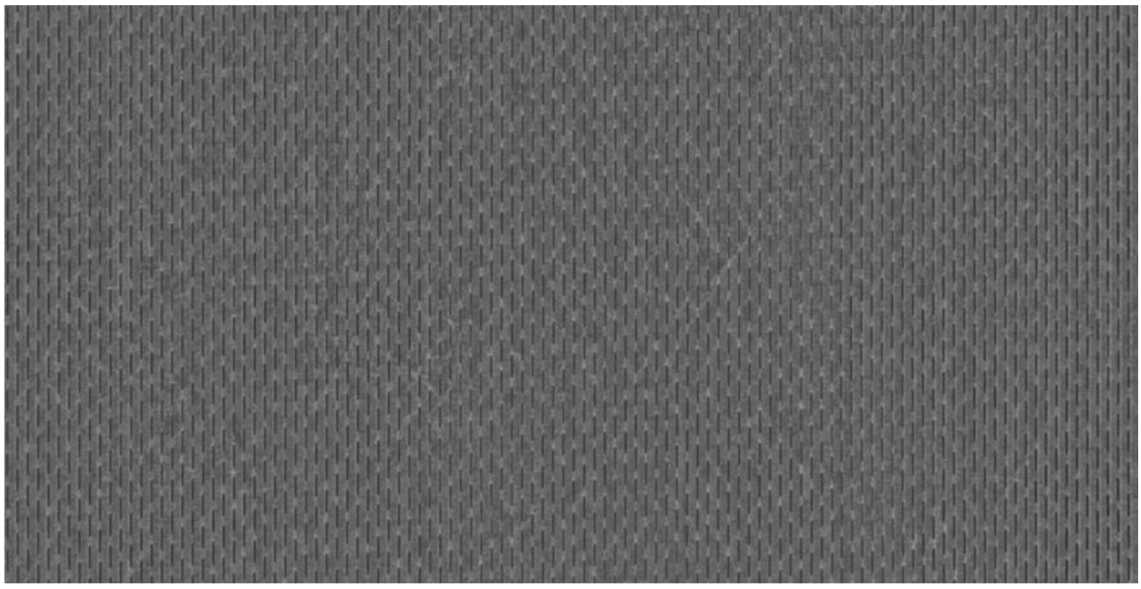
Na těchto originálních snímcích byly provedeny tři úpravy.

První úpravou je rozdělení snímku na menší snímky o velikosti 400x400. Důvodem je navýšení počtu snímků použitých při učení. Toto zajišťuje Matlabový skript s názvem *Rozdělení.m*. Tímto způsobem byl získán dataset o velikosti 3310 snímků (a v každém snímku více než 90 vzorů otisku kalandru), což se ukázalo být jako dostatečné pro učení sítě.

Druhou úpravou byla ekvalizace obrazových dat. Na obrázku můžeme vidět, že se střídají světlé a tmavé části netkané textilie s tím, jak lokálně kolísá gramáž. To ztěžuje proces učení, protože takový snímek je mnohem složitější. Byl tedy navržen algoritmus, pomocí kterého došlo k vyrovnání lokálních jasových hodnot ve snímku. Jednalo se o okno, které bylo posouváno po celém obraze a v tomto okně byl počítán jasový medián. Velikost okna byla zvolena tak, aby se do něho vešla gravura, zároveň však zabírala menší část snímku a neovlivnila tak výpočet jasového mediánu. Pro další výpočet byla využitá konstantní jasová hodnota (v tomto případě hodnota 100), na kterou bylo potřeba převést celý snímek. Dále se vypočítal podíl této konstantní jasové hodnoty a mediánu aktuálně analyzovaného okna. Výsledkem bylo číslo, kterým byla vynásobena hodnota každého pixelu v okně. Tyto hodnoty bylo uloženy do nového snímku na odpovídající místo. Algoritmus realizuje Matlabový skript s názvem *Ekvalizace.m*. Výsledek ekvalizace pro snímek, který je na obrázku 5.14, je na obrázku 5.15.



Obrázek 5.14 Příklad výřezu originálního snímku.



Obrázek 5.15 Ekvalizovaný originální snímek.

Poslední úprava se týkala pouze testovacích dat, která byla rozdělena na dvě části. První část dat obsahovala 40 snímků, ve kterých chybí na jednom místě gravura. Tato gravura byla odstraněna ručně na náhodných místech. Druhá část obsahovala 77 snímků, které byly v pořádku.

Použité snímky byly nahrány na Google Drive, odkud byli načteny do Google Colab pro další zpracování.

5.3.2 Nahrání snímků a vytvoření datasetu

Snímky byly nahrány z jednotlivých složek do tensorů, které měli rozměr (*počet snímků, počet kanálů, řádky a sloupce)* a následně na nich byly provedeny úpravy.

První úpravou byl převod snímku do grayscale podoby, z důvodu menšího množství dat, které bylo potřeba zpracovat. Dále tato data byla normalizována do rozsahu 0 až 1. Tyto dva tensory o rozměrech (47,1,1000,6105) a (50,1,1000,7823) byly využity pro vytvoření snímků o rozměrech 400x400.

Výsledný tensor má tedy rozměr (3310,1,400,400) a obsahuje všechna data určená pro testování a validaci. Tento tensor byl dále rozdělen pomocí funkce na trénovací a testovací data v poměru 0.8:0.2.

```
x_train, x_test = train_test_split(data, train_size = 0.8)
```

Trénovací a testovací data byla následně využita pro vytvoření dataloaderů, pomocí nichž je efektivnější trénování neuronové sítě. Jako velikost dávky *Batch* byla zvolena hodnota 32, která se při testování ukázala být vhodná.

```
train_dataLoader = DataLoader(dataset = x_train,
                              batch_size = BATCH_SIZE,
                              shuffle = True
)
test_dataLoader = DataLoader( dataset = x_test,
                              batch_size = BATCH_SIZE,
                              shuffle = False
)
```

5.3.3 Vytvoření modelu variačního autoenkodéru

Následně byla vytvořena architektura variačního autoenkodéru, která byla zvolena na základě experimentování s různým nastavením, variantami a parametry jednotlivých komponent. Vybrána byla ta, která dosahuje nejlepších výsledků.

Tyto komponenty jsou definovány v konstruktoru třídy *VariationalAutoencoder*. Konkrétní počet vrstev a jejích parametrů se zvolil na základě testování a vyhodnocení různých možností. Byly vybrány vrstvy a parametry vrstev, které mají nejmenší hodnoty ztrátové funkce. Kód je vidět níže.

```
def __init__(self, input_dim = 1, featureDim = 50*50*64, zDim = 500):
    super().__init__()
    self.Conv1 = nn.Conv2d(in_channels = input_dim, out_channels = 16,
 kernel_size = 3,padding = 1, stride = 1)
    self.Conv2 = nn.Conv2d(in_channels = 16,out_channels = 32, kernel_
size = 3,padding = 1, stride = 1)
    self.Conv3 = nn.Conv2d(in_channels = 32,out_channels = 64, kernel_
size = 3,padding = 1, stride = 1)

    self.Lin1 = nn.Linear(featureDim, zDim)
    self.Lin2 = nn.Linear(featureDim, zDim)
    self.Lin3 = nn.Linear(zDim, featureDim)

    self.IConv1 = nn.ConvTranspose2d(64, 32, 3,padding = 1, stride = 1
)
```

```
    self.IConv2 = nn.ConvTranspose2d(32, 16, 3,padding = 1, stride = 1
)
    self.IConv3 = nn.ConvTranspose2d(16, input_dim, 3,padding = 1, str
ide = 1)

    self.relu = nn.ReLU()
    self.pol = nn.MaxPool2d(2)
    self.ipol = nn.Upsample(scale_factor=2, mode='nearest')
```

Dále byly definovány metody třídy, ze kterých se variační autoenkodér skládá.

Enkodér je popsán na kódu níže, má tři konvoluční bloky. Každý konvoluční blok obsahuje konvoluční vrstvu, ReLU a MaxPooling vrstvu. Každý blok tedy provede operaci konvoluce, ReLU a zmenší dimenzi na polovinu. Dále obsahuje dvě plně propojené vrstvy, které jsou výstupem a reprezentují proměnné `mu` a `logVar`. Pro přechod z konvoluční vrstvy, kde jsou vstupem 2D data, do plně propojených vrstev, kde vstupem mohou být pouze 1D data, se využívá funkce `view()`.

```
  def encoder(self,x):
    x = self.relu(self.Conv1(x))
    x = self.pol(x)
    x = self.relu(self.Conv2(x))
    x = self.pol(x)
    x = self.relu(self.Conv3(x))
    x = self.pol(x)
    x = x.view(-1, 50*50*64)
    mu = self.Lin1(x)
    logVar = self.Lin2(x)
    return mu, logVar
```

Následně je definována metoda reparametrizace, výstupem je vektor z, který bude vstupem pro dekodér. Kód je níže.

```
  def reparameterize(self, mu, logVar):
    std = torch.exp(logVar/2)
    eps = torch.randn_like(std)
    return mu + std * eps
```

Poslední metodou je dekodér, jeho vstupem je vektor z, ze kterého se bude rekonstruovat původní snímek. Obsahuje tři konvoluční bloky, kde v každém bloku se vykoná transponovaná konvoluce, ReLU a UpSampling. Postupně se tedy rozměr bude zvětšovat až na původní velikost. Kód je níže.

```
  def decoder(self, z):
    x = self.relu(self.Lin3(z))
    x = x.view(-1, 64, 50, 50)
    x = self.ipol(x)
    x = self.relu(self.IConv1(x))
    x = self.ipol(x)
    x = torch.relu(self.IConv2(x))
    x = self.ipol(x)
    x = torch.sigmoid(self.IConv3(x))
```

```
    return x
```

Poslední je metoda *forward()*, která definuje posloupnost dopředného šíření vstupních dat. Kód je níže.

```
  def forward(self, x):
    mu, logVar = self.encoder(x)
    z = self.reparameterize(mu, logVar)
    out = self.decoder(z)
    return out, mu, logVar
```

5.3.4 Inicializace modelu a hyperparametrů

Následně je model variačního autoenkodéru inicializován a převeden na GPU výpočetní jednotku, pokud je k dispozici.

Jako optimalizační algoritmus byl zvolen Adam s learning rate 0.001, který byl zvolen na základě otestování různých možností a sledování vlivu na velikost trénovací a testovací chyby.

Dále jako ztrátovou funkci byla vybrána MSELoss, která porovnává jednotlivé pixely. Celková hodnota ztrátové funkce tedy závisí na podobnosti dvou porovnávaných snímků. Pro co nejvíce věrohodnou rekonstrukci je cílem zajistit co nejmenší hodnotu této funkce.

Nakonec byl definován počet iterací (epochs), během kterých model zpracuje celý dataset. V tomto případě byla zvolena hodnota 25, při větší hodnotě už nedochází k výraznému snižování hodnoty ztrátové funkce, což znamená, že se model více neučí.

```
model_0 = VariationalAutoencoder()
model_0 = model_0.to(device)
optimizer = torch.optim.Adam(model_0.parameters(), lr = 0.001)
loss_fn = torch.nn.MSELoss()
epochs = 25
```

5.3.5 Trénování modelu

Učení probíhá ve for cyklu, který bude mít počet iterací roven hodnotě *epochs*. Na konci každého cyklu dojde k vytištění aktuální trénovací a validační hodnoty ztrátové funkce. Zároveň dochází k ukládání hodnot trénovací i testovací ztrátové funkce, aby je bylo možno při vyhodnocení vynést do grafu.

V jedné iteraci dojde k trénování i validaci sítě.

Nejdříve probíhá trénování sítě. To znamená, že se model nastaví do trénovacího módu a dále dochází k načtení dávek snímků z dataloaderu. For cyklus má délku počtu dávek v dataloaderu.

Dále proběhne výpočet ztrátové funkce, která se sčítá mezi iteracemi. Na konci jedné iterace trénování je podělena počtem dávek v dataloaderu. Tím se vypočítá průměrná hodnota ztrátové funkce jedné iterace trénování.

Následně se zavolá funkce, která vynuluje gradienty, aby tyto gradienty mohly být následně nastaveny pomocí algoritmu zpětného šíření chyby. Dále se vypočte

gradient pro každý parametr. Nakonec dojde k aktualizaci jednotlivých parametrů na základě vypočtených gradientů a vytisknutí průběhu zpracovaných dávek z dataloaderu. Kód se nachází níže.

```
  for batch ,(X) in enumerate(train_dataLoader):
    model_0.train()
    X_pred,mu,sigma = model_0(X)
    loss = loss_fn(X_pred,X)
    train_loss += loss
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    if batch % 10 == 0:
      print(f"Batch {batch* len(X)} / {len(train_dataLoader.dataset)}
samples")
```

5.3.6 Validace modelu

Validace modelu se provede následně po trénování modelu.

Nejdříve je potřeba přepnout model do validačního módu. Následně dojde k postupnému načítání dávek snímků z dataloaderu a vložení na vstup modelu. Dále se vypočte ztrátová funkce, která se sčítá mezi jednotlivými iteracemi. Po skončení cyklu dojde k vydělení ztrátové funkce počtem dávek a uložení pro další zpracování. Příklad kódu je níže.

```
model_0.eval()
  with torch.inference_mode():
    for X_test in test_dataLoader:
       test_pred,sigma,mu = model_0(X_test)
       loss = loss_fn(test_pred,X_test)
       test_loss += loss
    test_loss /= len(test_dataLoader)
    testing_loss[epoch] = test_loss
```

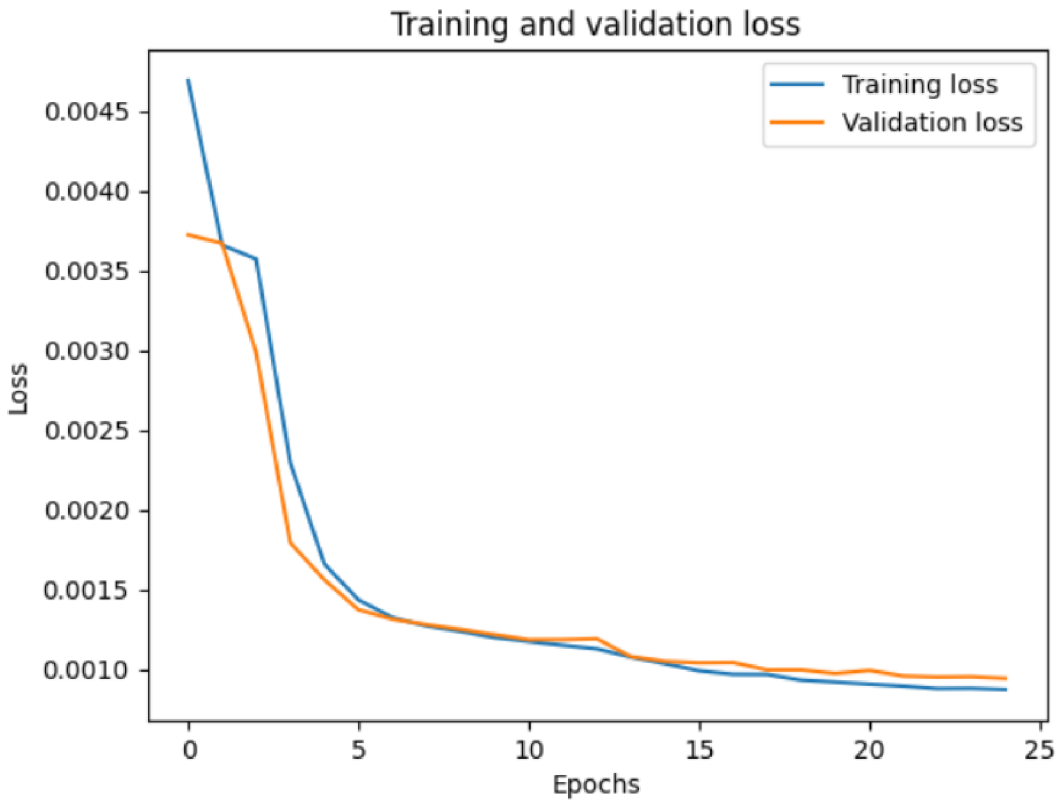
5.3.7 Průběh trénovací a validační ztrátové funkce

Po dokončení celého cyklu trénování jsou hodnoty ztrátové a validační funkce vyneseny do jednoho grafu, který je na obrázku 5.16. Z průběhu je vidět, že u tohoto modelu nedochází k overfittingu, protože obě hodnoty ztrátové funkce jsou ke konci cyklu podobné. Zároveň nedochází k underfittingu, protože obě hodnoty se drží na nízkých hodnotách.

Hodnota validační ztrátové funkce ke konci cyklu téměř neklesá, takže lze říci, že při využití delšího cyklu učení (větší číslo *epochs*), by nedošlo k lepšímu učení modelu. Při delším učení by pravděpodobně postupně začalo docházet k overfittingu.

Je vidět velký rozdíl na začátku cyklu mezi hodnotou trénovací a validační ztrátové funkce. To je způsobeno tím, že hodnota validační ztrátové funkce se počítá až po kompletním trénování v rámci jednoho cyklu. Tím pádem trénovací chyba má ze začátku velkou hodnotu a snižuje se až postupně, zároveň však tyto vysoké hodnoty

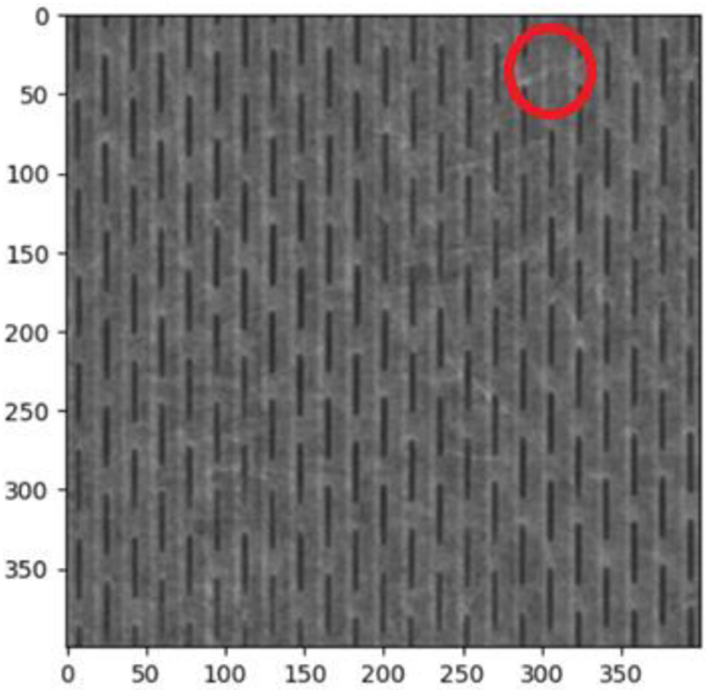
jsou zahrnuty do výpočtu celkově trénovací chyby jednoho cyklu. Validační chyba je však vypočítaná, až dojde k určitému natrénování sítě, proto ze začátku má nižší hodnoty.



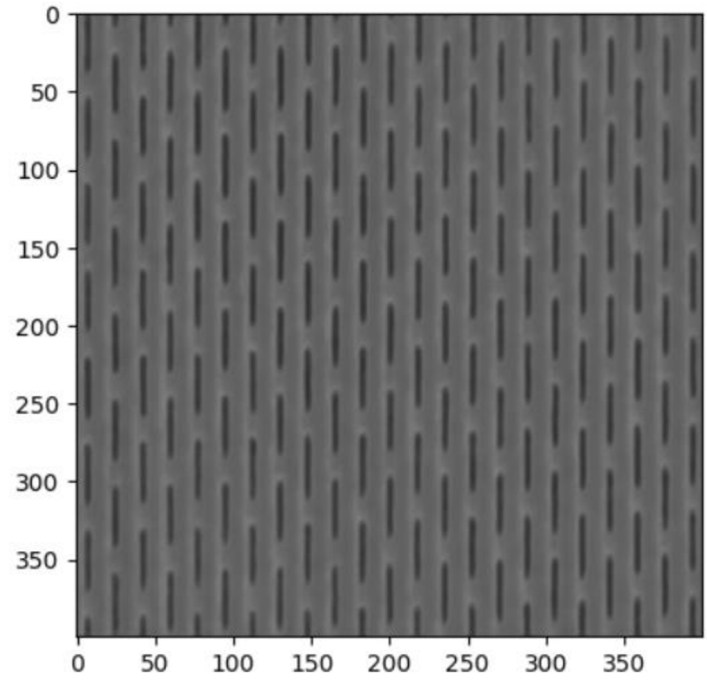
Obrázek 5.16 Průběh testovací a validační ztrátové funkce

5.3.8 Rekonstrukce modelu

Pro lepší představu o funkčnosti modelu budou následně zobrazeny výsledky natrénovaného modelu. Na obrázku 5.17 je vstup modelu, jedná se o anomální snímek o velikosti 400 x 400.



Obrázek 5.17 Vstupní snímek modelu s vyznačenou anomálií



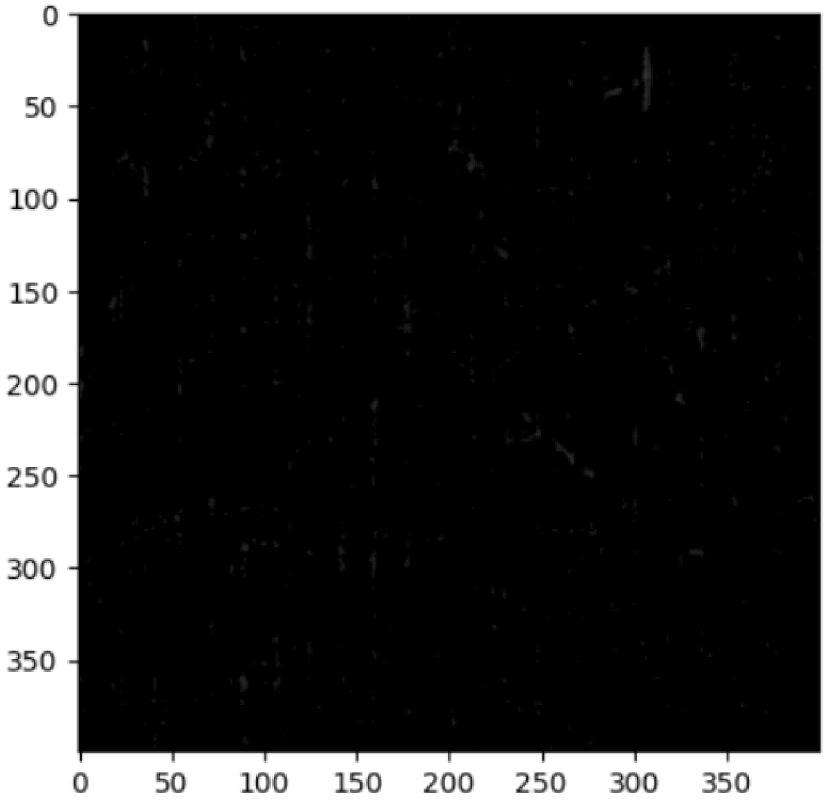
Obrázek 5.18 Výstupní snímek modelu

Na obrázku 5.18 je výstup modelu. Jedná se o rekonstruovaný snímek, na kterém je vidět, že model je schopen správně rekonstruovat naučený vzor a v případě míst, kde tento vzor chybí, ho doplnit. Dále je vidět, že model už není schopen dokonale

rekonstruovat netkanou textilií, což je očekávaná vlastnost, jelikož je spíše nepravidelná.

5.3.9 Rozdíl vstupního a výstupního

Následně je proveden rozdíl vstupního a výstupního snímku a prahování, aby se odfiltrovalo co nejvíce šumu. Tím se získá snímek, na kterém jsou vidět anomálie. To je vidět na obrázku 5.19. Zde je zřetelně vidět oblast, kde chybí gravura. Tato největší oblast má zároveň specifický tvar, který odpovídá gravuře. Těchto dvou vlastnosti bylo proto využito pro další zpracování.



Obrázek 5.19 Rozdíl vstupního a výstupního snímku

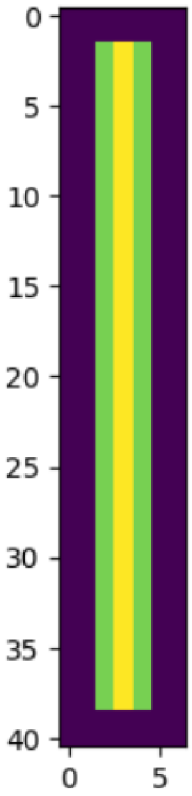
5.3.10 Detekce anomálie

V posledním zpracování tedy zbývá analyzovat, zda se na snímku nachází anomálie, a pokud ano, nalézt polohu této anomálie. Toto už není součástí neuronové sítě, zde se využívá knihovny pro zpracování obrazu a počítačové vidění, OpenCV.

5.3.11 Využití funkce cv::MatchTemplate()

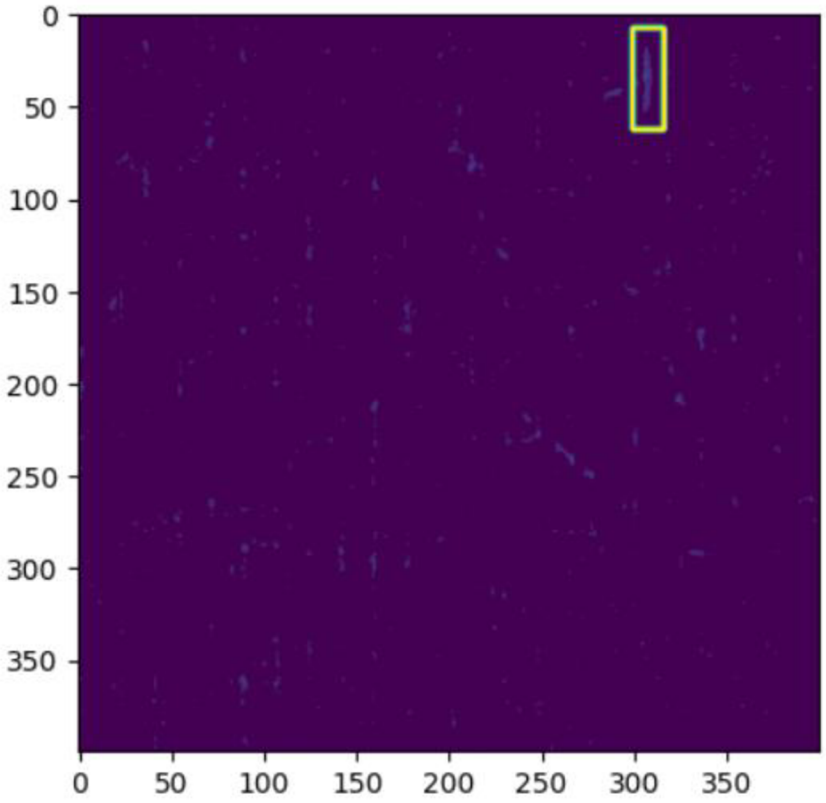
První otestovanou metodou hledání anomálie bylo využití knihovny OpenCV a funkce `cv::MatchTemplate()`, která počítá míru shody šablony s aktuálně porovnávanou oblastí na snímku.

Pro tento účel byla vytvořena pomocí Matlabového skriptu *šablona.m* šablona, která velikostí a intenzitou odpovídá chybějící gravuře. Pro lepší přehlednost je využitá barevná mapa. Šablona je na obrázku 5.20. V místech, kde je největší shoda šablony a výřezu snímku, je na výstupním snímku největší hodnota. Pro určení, zda je oblast anomální, lze tedy využít práh, jehož velikost byla nastavena na základě testování více snímků. V případě, kdy je na každém snímku maximálně jedna anomálie, vyhledá se místo s maximální hodnotou.



Obrázek 5.20 Šablona

Pokud bude vyhodnoceno, že se na snímku nachází anomálie, pak na základě polohy této anomálie se nakreslí do snímku obdélník okolo této anomálie. Příklad je na snímku 5.21.



Obrázek 5.21 Detekovaná anomálie

5.3.12 Využití Contours

Na obrázku 5.19 je vidět, že se na snímku nachází uzavřené oblasti. Zároveň je vidět, že místo anomálie má specificky tvar, který vychází z původního tvaru gravury. Byla tedy otestována i možnost využiti `cv::Contours`.

Pomocí funkce `cv.findContours()` se naleznou všechny uzavřené oblasti ve snímku.

Dále se pracuje s předpokladem, že Contour s největší oblastí bude odpovídat právě anomálií. Pro zpřesnění se využije dalších výpočtů, kde se vypočítá rozptyl hodnot v x souřadnici a y souřadnici této Contour. Pokud Contour má tyto dvě hodnoty rozptylu v mezích, které odpovídají velikosti gravury, pak jí lze pokládat za anomálii gravury. Pro lepší přehlednost je zvýrazněna stejně jako v předcházející metodě.

6. VYHODNOCENÍ METOD

V této kapitole budou vyhodnoceny všechny navrhnuté metody. Budou analyzovány na základě robustnosti a doby trvání.

6.1 Statistická metoda

Jednalo se o první navrhnutou a implementovanou metodu, kde princip se zdál být jednoduchý a intuitivní.

6.1.1 Robustnost

Z hlediska robustnosti tato metoda nevyniká. Pro správnou funkčnost této metody je potřeba mít jasně nastavené parametry na základě snímané textilie (pixelové rozlišení, jasová hodnota).

Jako první je potřeba mít správně nastavenou velikost výřezu, který se posouvá po obraze a v němž se počítá počet bílých pixelů. Pokud bude větší nebo menší, než je optimum, metoda nebude fungovat správně. Velikost výřezu bude záviset taktéž na typu gravury, která se nachází na netkané textilii.

Dále bude nutno nastavit správnou hodnotu stride, která vyjadřuje velikost posunutí elementu. Pokud bude stride hodnota 1, potom se výřez bude posouvat v obraze vždy o hodnotu své velikosti. Mohlo by se tedy stát, že anomálie by nebyla zaznamenána, proto by bylo lepší využít stride alespoň 2, aby se dva po sobě jdoucí výřezy překrývaly.

Dále je důležitá správná hodnota prahu, pomocí které získáme binární snímek, kde je gravura bílá a pozadí černé. Hodnota prahu bude záviset na jasových vlastnostech netkané textilie.

Pokud však dojde k správnému nastavení všech těchto parametrů, potom tato metoda se ukázala být funkční pro detekci anomálií gravury na netkané textilii.

6.1.2 Čas

Dále byla měřena doba vykonání jednoho cyklu. Tedy od načtení snímku, po zpracování a konečné vytisknutí polohy anomálie. Bylo měřeno více cyklů a výsledná doba pro měření byla získána průměrnou hodnotou.

Průměrná doba pro zpracování jednoho snímku vychází na hodnotu 125 ms. Je potřeba zmínit, že tato hodnota je vztažena k výpočetnímu zařízení a jeho využitých komponent. V tomto případě se jednalo procesor AMD Ryzen 4000.

Pokud se bude nacházet vada na kalandrovacím válci, potom tato vada se bude periodicky vyskytovat na netkané textilii. Dostatečné by mohlo být jednou za čas získat snímky netkané textilie, které odpovídají například pěti otočkám válce a analyzovat, zda se na těchto snímcích anomálie nachází.

Z toho lze vidět, že využití by našla v případech, ve kterých by stačilo získat snímky a tyto snímky by se analyzovaly následně bez požadavků na real-time zpracování.

Pokud by však zákazník požadoval, bylo by možno ji využít i pro real-time zpracování, jelikož analýza netkaných textilií se v praxi provozuje při nižších rychlostech pohybu textilie.

.

6.2 Fourierova transformace

Původní hypotéza, že úloha by mohla být řešena pouze pomocí Fourierovy transformace, se ukázala být chybná. Porovnáním frekvenčního spektra správného a anomálního snímku se nezíská žádná informace o poloze anomáliích v obraze. Tím pádem další vývoj této metody nepokračoval. Následný vývoj byl proto zaměřen na poslední a nejvíce nadějnou metodu, která je popsána dále.

6.3 Neuronové sítě

Jedná se o poslední implementovanou metodou, od které byly očekávány nejslibnější výsledky.

6.3.1 Robustnost

Tato metoda se ukázala být dostatečně robustnou pro požadovanou aplikaci za splnění podmínky, že analyzované snímky budou mít podobné vlastnosti (jas, typ gravury a velikost) jako snímky, na kterých byla síť trénována.

6.3.2 Přesnost

Pro vyhodnocení výstupu neuronové sítě a následné detekce anomálií byla využitá knihovna OpenCV a její funkce. Byly implementovány dva typy detekce anomálií. První využívá hledání největších Contours a vyhodnocení jejich vlastností, druhá hledání šablony. Pro vyhodnocení bylo využito 40 snímků, na kterých se nachází ručně vytvořená anomálie a 77 snímků, na kterých se anomálie nenachází. Pro obě metody byla vytvořena matice záměn. Pozitivní vyjadřuje, že je bez anomálie, negativní, že na snímku je anomálie.

Matice záměn pro detekci anomálii pomocí hledání a analýzy největší Contour ve snímku se nachází v tabulce 6.1.

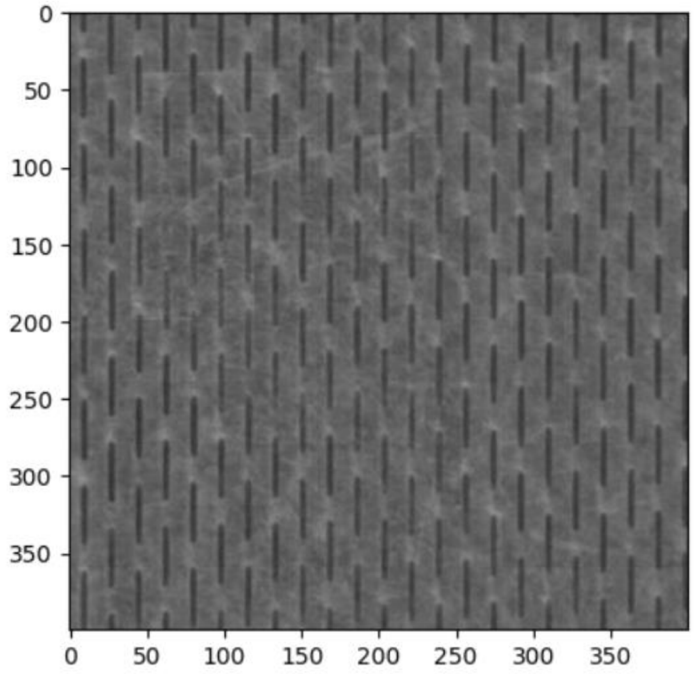
		Predikce	
Skutečnost		Positive	Negative
	Positive	76 TP	1 FN
	Negative	14 FP	26 TN

Tabulka 6.1 Matice záměn pro vyhodnocení metody detekce Contours

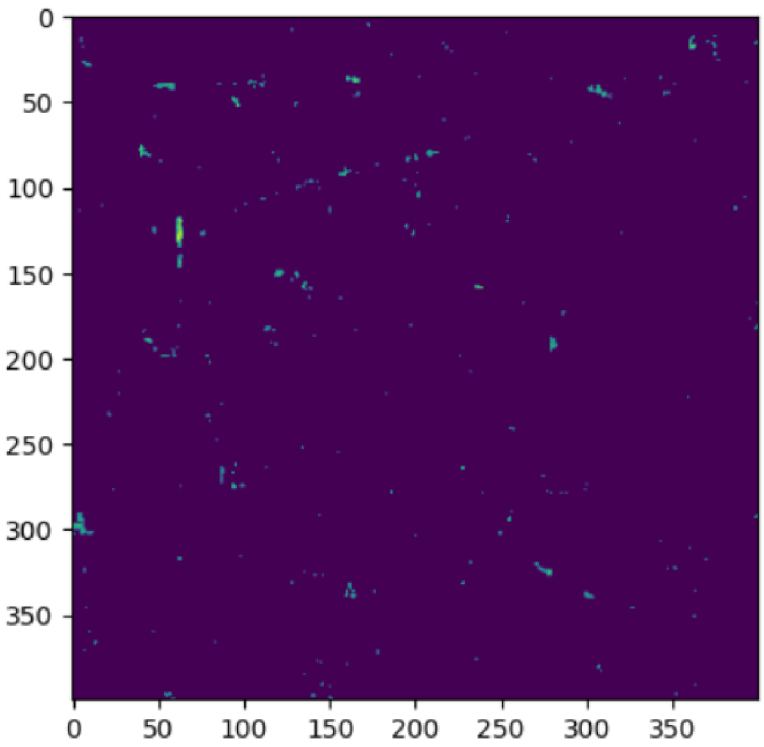
Z této tabulky je vidět, že tato metoda je dostatečně přesná v klasifikaci snímků, na kterých se anomálie nenachází. Avšak přesnost této metody v případě snímku, na kterých se anomálie nachází, už tak vysoká není. Na základě rovnic (3.1),(3.2) a (3.3) byly vypočítány další vlastnosti klasifikace.

Celková správnost se rovná 87 %, senzitivita 98.7 % a specifita 65 %.

Příklad snímku, na kterém algoritmus nefunguje správně, je na obrázku 6.1 a výsledek algoritmu na obrázku 6.2. Je vidět, že ačkoliv pro lidské oko je stále jasně zřetelné, kde se anomálie nachází, pro algoritmus hledání Contour o největší velikosti anomálie nebyla nalezena. Důvodem je to, že anomálie je rozdělena na dvě části, tím pádem nemá vlastnosti, které odpovídají velikosti a rozměrům gravury. Z tohoto důvodu metoda detekce selhává u tohoto počtu snímků. Proto byla hledána a implementována metoda, pomocí které se dosáhne lepších výsledků.



Obrázek 6.1 Originální snímek s anomálií



Obrázek 6.2 Výsledek algoritmu pro detekci anomálií

Matice záměn pro detekci anomálii pomocí hledání šablony chybějící gravury ve snímku se nachází v tabulce 6.2. Z této tabulky je vidět, že tato metoda je mnohem přesnější pro detekování anomálií. Avšak je o něco snížená přesnost u snímků, kde se anomálie nenachází v porovnání s předcházející metodou. Na základě rovnic (3.1),(3.2) a (3.3) byly vypočítány další vlastnosti klasifikace.

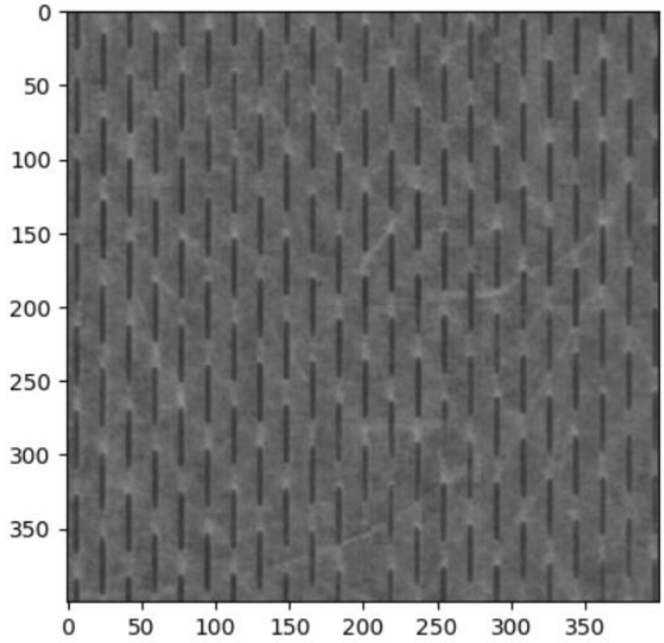
Celková správnost se rovná 94.8 %, senzitivita 96 % a specifita 92.5 %.

	Predikce		
Skutečnost		Positive	Negative
	Positive	74	3
	Negative	3	37

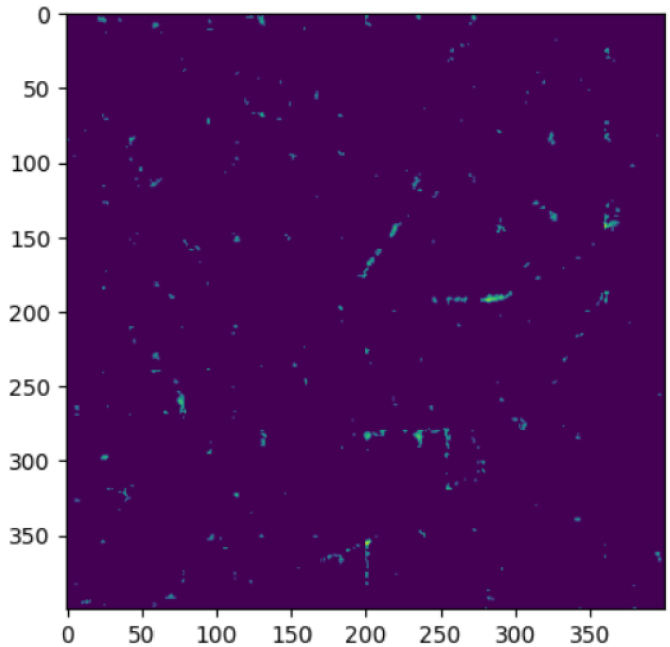
Tabulka 6.2 Matice záměn pro vyhodnocení metody detekce pomocí šablony

Příklad snímku, na kterém algoritmus nefunguje správně, je na obrázku 6.3 a výsledek algoritmu na obrázku 6.4. Je vidět, že ačkoliv pro lidské oko je stále ještě rozpoznatelné, kde se anomálie nachází, pro algoritmus hledání šablony anomálie nebyla nalezena. Důvodem je to, že anomálie je nesouvislá, tím pádem shoda mezi

šablonou a oblastí není dostatečná. Z tohoto důvodu v tomto i zbývajících dvou špatně detekovaných snímcích selhává. Každopádně je efektivnější pro hledání anomálií než metoda hledání Contour, jelikož dokáže mnohem lépe detekovat anomálie, které nejsou spojité, ale stále jsou dobře rozeznatelné.



Obrázek 6.3 Originální snímek s anomálií



Obrázek 6.4 Výsledek algoritmu pro detekci anomálií

6.3.3 Čas

Obě metody klasifikace byly porovnány z hlediska časové náročnosti zpracování jednoho snímku. Dále vypočet probíhal v Google Colab, proto jako výpočetní zdroje byli využity zdroje Googlu. Byl získán čas v případě výpočtu s využitím pouze CPU a v případě použití akcelerátoru GPU. Pro získání jedné hodnoty byla využita průměrná hodnota zpracování jednoho snímku. Hodnoty se nachází v tabulce 6.3. Je vidět, že v obou případech je metoda Contours výpočetně méně náročná. Dále je vidět, že je celkem výrazný rozdíl mezi využitím CPU a akcelerátoru GPU. V případě využití akcelerátoru GPU by neměl být problém s real-time zpracováním, pokud by si to zákazník vyžadoval.

	CPU	GPU
Contours	185 ms	8.7 ms
Šablona	241 ms	12.9 ms

Tabulka 6.3 Porovnání čau zpracování snímku různými metodami

7. ZÁVĚR

Tato diplomová práce se zabývá texturní analýzou netkané textilie, která prošla procesem kalandrování, při kterém se na nich zároveň vytvořila gravura. Může se však stát, že v důsledku poškození kalandrovacího válce se gravura nepřenese na netkanou textilii správně. Detekce takové vady je pak pro člověka nadlidský úkol, jelikož rozměry kalandru mohou být velmi malé. Úkolem této práce je detekovat anomálie na netkané textilií efektivněji pomocí nástrojů Počítačového vidění a neuronových síťí.

V první části je popsán proces výroby netkaných textilií, jejich úpravy a proces kalandrování z pohledu průmyslu. Čerpáno je převážně od dodavatele Edana, který se věnuje vytváření netkaných textilií, jejich úprav a kalandrování.

V druhé části je popsána teoreticky Fourierova transformace, která se následně ukázala být významným nástrojem pro následné vypracování zadání. Pomocí ní lze odfiltrovat jednotlivé frekvence ve snímku. Tak lze získat snímek, který je následně mnohem jednodušší analyzovat a využít pro detekci anomálie.

V třetí části jsou popsány neuronové sítě a její části.

Ve čtvrté části je popsána architektura variačního autoenkodéru, který byl využit pro řešení úlohy.

V páté kapitole je popsáno řešení úlohy pro jednotlivé metody.

Nakonec je vytvořeno zhodnocení jednotlivých metod.

V této práci byly navrženy a otestovány různé přístupy řešení úlohy. První metodou bylo využití Fourierovy transformace s následným statistickým zpracováním. Na základě pokusu s anomálním snímkem se ukázalo, že tento přístup řešení je možný. U této metody však je potřeba nastavit správně parametry algoritmu, aby fungoval správně. Proto je možné využití pouze v případě, kdy se podmínky (nasvícení, vzdálenost kamery od textilie) nebudou v čase výrazně měnit. Z hlediska časové náročnosti zpracování jednoho snímku, by měla být tato metoda použitelná v případě, kdy by byl požadavek od zákazníka na real-time zpracování. Důvodem je to, že kontrola výrobku probíhá při vyšších hodnotách gramáže, a tedy nižší rychlosti výroby.

Druhou metodou bylo využití pouze Fourierovy transformace, která ve výsledku nepřinesla kýžený výsledek a není tedy funkční pro tuto úlohu.

Nakonec byla implementována metoda, která se zdála být nejvhodnější pro tuto úlohu. Byla vytvořena neuronová síť, v níž byl její architekturou zvolen variační autoenkodér. Na základě snímků, které byly dodány a předzpracovány do formy vhodného jako dataset sítě, byla tato síť naučena komprimovat vstupní snímky do mnohem menšího rozměru a z tohoto menšího rozměru rekonstruovat zpět do původní podoby s co nejmenší odchylkou. Parametry sítě byly laděny postupně a vybrána nejvhodnější konfigurace, která zajišťovala nejlepší výsledky. Následně bylo proveden rozdíl vstupního a výstupního snímku. Dále byly implementovány dvě metody pro detekci přesné polohy anomálie ve snímku a otestovány.

Z hlediska přesnosti vychází lépe metoda detekce pomocí šablony, která dokázala detekovat správně 37 ze 40 anomálních snímků. Ačkoliv v případě snímků, na kterých žádná anomálie nebyla, detekovala 3 anomálie ze 77, v praxi se každá anomálie ověřuje člověkem, a je tedy důležitější detekovat snímky s anomáliemi. Proto tedy metoda detekce šablony je vhodnější.

Z hlediska časové náročnosti vychází lépe metoda Contours, avšak rozdíl není tak výrazný oproti metodě šablony. Výrazný rozdíl je však mezi využitím pouze CPU a GPU. V případě GPU je rychlost zpracování kolem 10 ms. Vzhledem k tomu, že ke kontrole celistvosti kalandru dochází vždy při vyšší gramáži (a tedy nižší rychlosti výroby), lze bezpečně použít i zpracování pomocí CPU a splnit tak požadavky na real-time zpracování v případě potřeby zákazníka.

LITERATURA

[1] SUNDARARAJAN, D. 2001. *The Discrete Fourier Transform: Theory, Algorithms and Applications. 1.* Singapore: World Scientific Publishing Co. Pte. Ltd. ISBN 981024521.

[2] RUSS, John C. *The image processing handbook*. 5th ed. Boca Raton: CRC/Taylor and Francis, c2007. ISBN 0-8493-7254-2. [Cit. 2022-12-25].

[3] *Kalandrování* [online]. [Cit. 2022-12-26]. Dostupné z URL: http://www.skolatextilu.cz/elearning/514/zaklady-textilnich-technologii/zuslechtovani-textilili/Kalandrovani.html

[4] *How are nonwovens made?* [online]. [Cit. 2022-12-29]. Dostupné z URL: https://www.edana.org/nw-related-industry/how-are-nonwovens-made

[5] *Nonwovens in daily life* [online]. [Cit. 2022-12-29]. Dostupné z URL: https://www.edana.org/nw-related-industry/nonwovens-in-daily-life

[6] *Fourierova transformace ve 2D* [online]. [Cit. 2022-12-29]. Dostupné z URL: https://www.uamt.fekt.vut.cz/~richter/vyuka/0910_mpov/tmp/integral_tr_2DFT.html

[7] *Jak funguje umělá inteligence?* [online]. [Cit. 2023-04-23]. Dostupné z URL: *https://azure.microsoft.com/cs-cz/resources/cloud-computing-dictionary/what-is-artificial-intelligence/#how*

[8] RUSSELL, Stuart J. a Peter NORVIG. *Artificial intelligence: a modern approach*. 3rd ed. Upper Saddle River: Prentice Hall, 2010. Prentice Hall series in artificial intelligence. ISBN 978-0-13-604259-4.

[9] *Neurons in Neural Network?* [online]. [Cit. 2023-04-23]. Dostupné z URL: https://www.baeldung.com/cs/neural-networks-neurons

[10] PANESAR, Arjun a Arjun PANESAR. *Machine Learning and AI for Healthcare: Big Data for Improved Heatlth Outcomes*. Coventry: Apress, 2019. ISBN 978-1-4842-3798-4.

[11] *The concept of Artifical neuron in Neural Network* [online]. [Cit. 2023-04-25]. Dostupné z URL: https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc

[12] *Activation functions in Neural Network* [online]. [Cit. 2023-04-25]. Dostupné z URL: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[13] *A gentle introduction to the ReLU* [online]. [Cit. 2023-04-25].
Dostupné z URL:
https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

[14] *Types of Neural networks [online].* [Cit. 2023-04-25]
Dostupné z URL:
https://www.mygreatlearning.com/blog/types-of-neural-networks/

[15] *Convolutional neural networks: an overview and application in radiology [online].* [Cit. 2023-04-26]
Dostupné z URL:
https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9

[16] *How to use Upsampling2D and Conv2DTranspose Layers in Keras [online].* [Cit. 2023-04-26]
Dostupné z URL:
https://machinelearningmastery.com/upsampling-and-transpose-convolution-layers-for-generative-adversarial-networks/

[17] *Understanding 3 most common loss function for Machine Learning Regression [online].* [Cit. 2023-04-26]
Dostupné z URL:
https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine-learning-regression-23e0ef3e14d3

[18] *Various optimizaton algorithms for training Neural network [online].* [Cit. 2023-04-29]
Dostupné z URL:
https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

[19] *Neural Networks: Forward pass and Backpropagation [online].* [Cit. 2023-05-01]
Dostupné z URL:
https://towardsdatascience.com/neural-networks-forward-pass-and-backpropagation-be3b75a1cfcc

[20] *Understanding Confusiob Matrix [online].* [Cit. 2023-05-01]
Dostupné z URL:
https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

[21] *What if PyTorch, and how does it works: All you need to know [online].* [Cit. 2023-05-01]
Dostupné z URL:
https://www.simplilearn.com/what-is-pytorch-article

[22] *Colaboratory – Frequently asked questions [online].* [Cit. 2023-05-01]
Dostupné z URL:
https://research.google.com/colaboratory/faq.html

[23] *Autoencoders Guide [online].* [Cit. 2023-05-01]
Dostupné z URL:
https://www.v7labs.com/blog/autoencoders-guide
[24] *Variational Autoencoders [online].* [Cit. 2023-05-01]
Dostupné z URL:
https://www.jeremyjordan.me/variational-autoencoders/
[25] *GPT-5: release date, claims of AGI, pushback, and more. [online].* [Cit. 2023-05-01]
Dostupné z URL:
https://www.digitaltrends.com/computing/gpt-5-rumors-news-release-date/

SEZNAM PŘÍLOH

PŘÍLOHA A - OBSAH PŘILOŽENÉHO DVD 72

Příloha A - Obsah přiloženého DVD

DVD se nachází v pevném krytu, který je přiložen k závěrečné práci. Obsahuje dvě hlavní složky.

A.1 Složka Dataset

Složka *Dataset* obsahuje snímky, které byli využity pro návrh Neuronové sítě.

Dále složka *Dataset* dále obsahuje dva Matlab skripty s názvy *Ekvalizace.m* a *Rozdělení.m*, pomocí kterých se snímky předzpracovali.

Snímky jsou rozděleny do složek na základě typu datasetu. V jedné složce jsou trénovací a validační data. Tato složka je dále rozdělena na původní dodané snímky, které jsou ve složkách *K1N* a *K2N* a jejich ekvalizované podoby, které jsou ve složkách *K1N_Ekvalizovane* a *K2N_Ekvalizovane*. Složka *Testovací data* obsahuje data, která byla využita pro testování. Ve složce *Puvodni* se nachází snímky, které prošli předzpracováním a ve složce *Ekvalizovane* jsou tyto snímky ekvalizované. Dále jsou v této složce snímky o velikosti 400x400 v původní podobě i ekvalizované. Každá z těchto složek obsahuje další dvě složky, kde se nachází snímky, na kterých je anomálie a snímky které jsou v pořádku.

Nakonec je zde složka *Vada*, kde se nachází dodaný snímek vady z průmyslu.

A.2 Složka Diplomka

Ve složce *Diplomka* se nachází software, který byl využit pro řešení úlohy a dokument této Diplomové práce s názvem *Dokument*.

Ve složce *Google Colab* se nachází dokument *Variacni_autoenkoder.ipynb*, ve kterém je implementována Neuronová síť. Dále se zde nachází souboru *model.pt*, který v sobě uchovává model s nastavenými parametry sítě, a dva soubory typu *.npy*, ve kterých je uložena hodnota testovací a validační ztrátové funkce.

Ve složce *Matlab* se nachází skripty a snímky, který byli využity pro metody využívající Fourierovy transformace a pro vytvoření šablony.