

V. Počítačová grafika a analýza obrazu

Update: 5. května 2018

Obsah

1	Osvětlovací modely a systémy barev v počítačové grafice.	2
2	Afinní a projektivní prostor. Afinní a projektivní transformace a jejich matematický zápis. Aplikace v počítačové grafice. Modelovací a zobrazovací transformace.	3
3	Křivky a plochy: teoretické základy (definice, rovnice, tečný a normálový vektor, křivosti, C_n a G_n spojitost), použití (Bézier, Coons, NURBS).	4
4	Geometrické a objemové modelování. Hraniční metoda, metoda CSG, výčet prostoru, oktantové stromy.	5
5	Standardní zobrazovací řetězec a realizace jeho jednotlivých kroků. Gouraudovo a Phongovo stínování. Řešení viditelnosti. Grafický standard OpenGL: stručná charakteristika.	6
6	Metody získávání fotorealistických obrázků (rekurzivní sledování paprsku, vyzařovací metoda, renderovací rovnice).	7
7	Kompresce obrazu a videa; principy úprav obrazu v prostorové a frekvenční doméně.	8
8	Základní metody úpravy a segmentace obrazu (filtrace, prahování, hrany).	9
9	Základní metody rozpoznávání objektů (příznakové rozpoznávání).	10

1 Osvětlovací modely a systémy barev v počítačové grafice.

- 2 Afinity a projektivní prostor. Afinity a projektivní transformace a jejich matematický zápis. Aplikace v počítačové grafice. Modelovací a zobrazovací transformace.

- 3 Křivky a plochy: teoretické základy (definice, rovnice, tečný a normálový vektor, křivosti, C_n a G_n spojitost), použití (Bézier, Coons, NURBS).

- 4 Geometrické a objemové modelování. Hraniční metoda, metoda CSG, výčet prostoru, oktantové stromy.

- 5 Standardní zobrazovací řetězec a realizace jeho jednotlivých kroků. Gouraudovo a Phongovo stínování. Řešení viditelnosti. Grafický standard OpenGL: stručná charakteristika.

- 6 Metody získávání fotorealistických obrázků (rekurzivní sledování paprsku, vyzařovací metoda, renderovací rovnice).

- 7 Komprese obrazu a videa; principy úprav obrazu v prostorové a frekvenční doméně.

8 Základní metody úpravy a segmentace obrazu (filtrace, prahování, hrany).

9 Základní metody rozpoznávání objektů (příznakové rozpoznávání).

9.1 Histogram orientovaných gradientů HOG

Základní myšlenkou je, že objekt v obraze může být pomocí vzhledu a tvaru charakterizován pomocí intenzity gradientů, i přestože neznáme jejich přesnou polohu v obraze. Autoři jsou N. Dalal a B. Triggs (2005).

1. před započítáním výpočtů je třeba normalizovat, například normalizaci barev a gamy, v případě černobílých obrázků k normalizaci kontrastu (tento krok může být přeskočen, dle Dalal a Triggs → předzpracování má malý vliv na výkon)
2. obraz se **rozdělí** na malé prostorové oblasti (buňky, například 8×8 pixelů)
3. pro každou buňku se vypočítá 1-D **histogram**, který je vypočítán ze všech pixelů z buňky (hodnoty buněk jsou rovnoměrně rozloženy do histogramu o 9 kanálech (binech) po 20° ; rozsah 0° – 180°) → výjde nám vektor o velikosti 9
4. buňky spojíme do větších **propojených bloků** 16×16 z důvodu normalizace osvětlení a kontrastu. Pro chodce se používá L2-norm normalizace, dle vztahu (1) → vznikne vektor velikosti $9 \times 4 = 36$ (čtyři 8×8 bloky)
5. vektor normalizovaných histogramů pro jeden blok nazýváme **deskriptor**.
6. spojíme tyto normalizované vektory do jednoho a získáme „trénovací“ vektor příznaků (features)

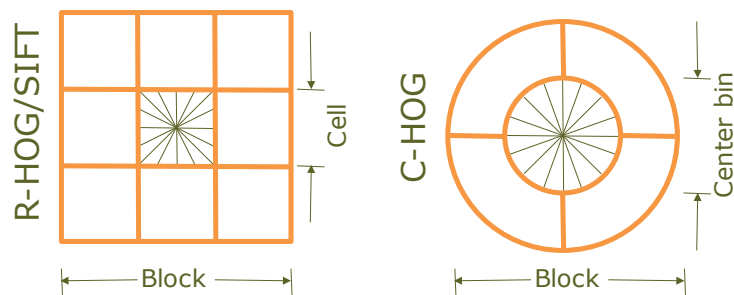
Existují dvě varianty spojení bloků, tzv. obdélníkové bloky (R-HOG) a kruhové bloky (C-HOG). Rozdělení do rozsahu 0 – 180° proto, že se jedná o bezznaménkové gradienty (unsigned) a bylo dokázáno, že fungují lépe než znaménkové (signed) 0 – 360° . Některé implementace HOG umožní určit, zda chceme používat signed gradienty.

$$\begin{aligned} L2 - norm : \quad f &= \frac{v}{\sqrt{\|v\|_2^2 + e^2}} \\ L1 - sqrt : \quad f &= \sqrt{\frac{v}{\|v\|_1 + e}} \end{aligned} \tag{1}$$

Nechť v je nenormalizovaný vektor obsahující všechny histogramy v daném bloku, $\|v\|_k$ je jeho k -norm pro $k = 1, 2$, a e je malá konstanta.

C-HOG (Kruhové HOG bloky) - lze nalézt ve dvou variantách: *s jedinou, centrální buňkou a úhlově rozdělenou centrální buňkou*. Dají se popsat čtyřmi parametry: počtem úhlů a radiálních kanálů (binů), poloměrem centrálního binu a faktorem roztažení pro poloměr dalších radiálních binů.

R-HOG (Obdélníkové HOG bloky) - tyto bloky jsou v praxi nejčastěji používané a reprezentují se třemi parametry: *počet buněk na blok, počet pixelů na buňku a počet binů (kanálů) na jeden histogram*. R-HOG bloky se také používají pro kódování informací.



Obrázek 1: Varianty geometrie spojení bloků

9.2 SIFT/SURF - detektory a popisovače klíčových bodů

Využívá se stejný princip tvoření histogramu jako u metody HOG

- klíčové body jsou nezávislé na osvětlení, velikosti, orientaci, pozici
- Octave - úroveň škálování
 - 4 oktávy a 5 rozmazání “ideál” dle prezentace, každá oktáva se 5x rozmáže
 - vypočítá se rozdíl mezi rozmazanými obrázky
 - klíčový bod najdeme jako minimum/maximum mezi různými urovněmi rozmazání
- poté musíme provést eliminaci slabých bodů
 - odebereme ty s malou intenzitou
 - odstraníme klíčové body, které leží na hraně - využít princip Harrisova detektoru hran - Hessian matice (matice druhých derivací)
- orientace bodu se vypočítá pomocí histogramu směrů gradientů v okolních bodech, zajišťuje nám to invarianci vůči rotaci (36 košů)
- Descriptor
 - 16x16 matice okolo klíčových bodů
 - rozdělit na 4x4 subbloky (16 bloků v 16x16 okolí)
 - * v nich vypočítat histogram orientací gradientu
 - * poté tento histogram převést do vektoru (viz HOG)
 - * spojit pro všechny bloky a máme výsledný vektor
- SURF má stejné kroky jako SIFT, jen má jiné „implementace“ kroků

9.3 Haarovy příznaky

- Na tomto přístupu je založen objektový detektor **Viola-Jones** (*Viola-Jones object detector framework*).
- Poskytuje v reálném čase **spolehlivou a konkurenceschopnou** detekci objektů.

- Může být vytrénován pro detekci různých objektových tříd (primárně určen pro **detekci obličejů**).
- Detektor pracuje s obrazy ve stupních šedi a skládá se ze tří částí. (Integrální obraz, Haar příznaků a AdaBoost algoritmus)

Integrální obraz je takový obraz (obrázek 4), kde každý bod x představuje součet hodnot předchozích pixelů doleva a nahoru. Spodní pravý bod obsahuje součet všech pixelů v obraze. Zápis integrálního obrazu je:

$$I(x, y) = \sum_{\substack{x' \leq x \\ y' \leq y}} i(x', y'),$$

kde $i(x', y')$ je hodnota pixelu na pozici (x, y) .

1	1	1
1	1	1
1	1	1

Obrázek 2: Vstupní obraz

1	2	3
2	4	6
3	6	9

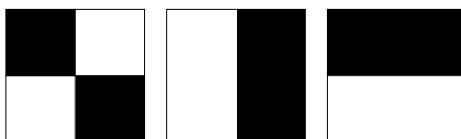
Obrázek 3: Integrální obraz

Obrázek 4: Převod obrazu na integrální obraz

Princip využití Haarových příznaků v obrazech je založen na pozorování, že lidská těla a obličeje mají některé podobné rysy. Právě tyto rysy mohou být porovnány pomocí Haarových příznaků. Jedná se například o tyto rysy:

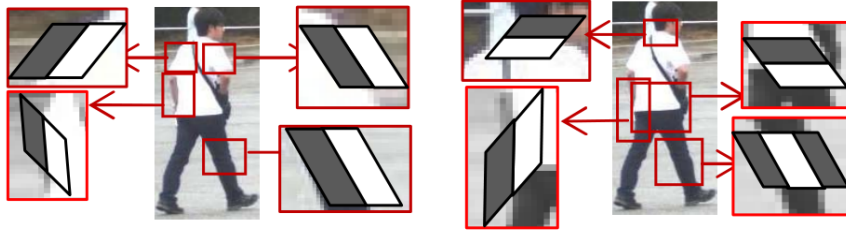
- Oční oblast je tmavší než oblast nosního mostu,
- hlava člověka je tmavší než její okolí,
- oblast mezi dolními končetinami je světlejší než samotné nohy.

Sada Haarových vlnek je na obrázku 5, jedná se pouze o základní sadu příznaků.



Obrázek 5: Základní sada Haarových příznaků

Pro identifikaci lidských postav se používá rozšířená sada vlnek, tzv. Haar-like příznaky. Klasifikační systém založený na těchto Haar-like příznacích dosahuje nižší falešně pozitivní detekce než původní Haar příznaky. Na obrázku 6 je příklad detekce pomocí Haar-like vlnek. Hodnota příznaku je rozdíl mezi sumou hodnot pixelů v bílé a černé oblasti Haarových vlnek.



Obrázek 6: Použití Haar-like příznaků na chodcích

9.4 LBP Lokální binární vzor

Hlavní myšlenkou LBP je, že struktury obrazu mohou být efektivně zakódovány porovnáním hodnot jednotlivých pixelů a jejich okolí. Tato metoda je odolná vůči jasovým změnám obrazu.

1. převod obrazu do stupňů šedi a jeho rozdělení do buněk
2. okolní hodnoty pixelů jsou porovnávány se středovým pixelem, pokud je jejich hodnota rovna nebo větší zapisuje se na tuto pozici jednička v opačném případě nula
3. tyto hodnoty seřadíme dle hodinových ručiček nebo naopak a získáme osmimístné binární číslo a převedeme do dekadické soustavy
4. z čísel, které jsme získali kombinací pixelů v buňkách, vypočítáme histogram
5. zřetězíme všechny histogramy buněk a získáme vektor příznaků pro celý obraz (jedná se o 256-dimenzionální vektor příznaků)

Matematicky lze LPB vyjádřit jako:

$$LBP_{P,R} = \sum_{p=0}^{P-1} 2^{s(p)} (g_p - g_c) 2^P, s(x) = \begin{cases} 1 & \text{pro } x \geq 0, \\ 0 & \text{pro } x < 0, \end{cases}$$

kde: P je počet bodů v okolí, R vyjadřuje vzdálenost bodů od středového pixelu, g_c je středový pixel, g_p je aktuální pixel.

Následující příklad se vztahuje k obrázku 10. Po porovnání pixelů se středovým pixelem jsme získali vzor 11110001. Tento vzor převedeme do dekadické soustavy a sečteme, $1 + 16 + 32 + 64 + 128 = 241$. Získali jsme hodnotu této buňky do vektoru příznaků.

6	2	2
7	6	1
9	8	7

Obrázek 7: *
Vstupní buňka

1	0	0
1		0
1	1	1

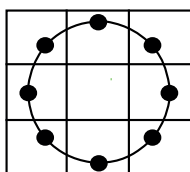
Obrázek 8: *
Prahové hodnoty

1	2	4
128		8
64	32	16

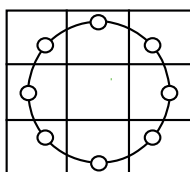
Obrázek 9: *
Pixely ohodnoceny váhou

Obrázek 10: Výpočet příznaku

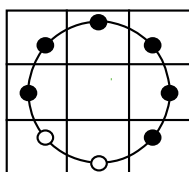
Výhoda této metody je její rychlý a snadný výpočet a odolnost vůči různým osvětlením. Na druhou stranu je těžší na trénování, protože výsledné dekadické číslo může mít obrovské množství možností (podle parametru P). K omezení lze využít uniformní vzory (obrázek 16). Pro parametr $P = 8$, získáme 59 vzorů.



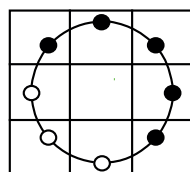
Obrázek 11: *
Bod



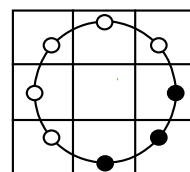
Obrázek 12: *
Bod/Plocha



Obrázek 13: *
Křivka



Obrázek 14: *
Roh



Obrázek 15: *
Hrana

Obrázek 16: Lokální okolí LBP metody

9.5 Klasifikátory

Klasifikace je obecný proces kategorizující objekty do určitých tříd. Termín klasifikátor někdy odkazuje také na matematickou funkci, implementovanou klasifikačním algoritmem.

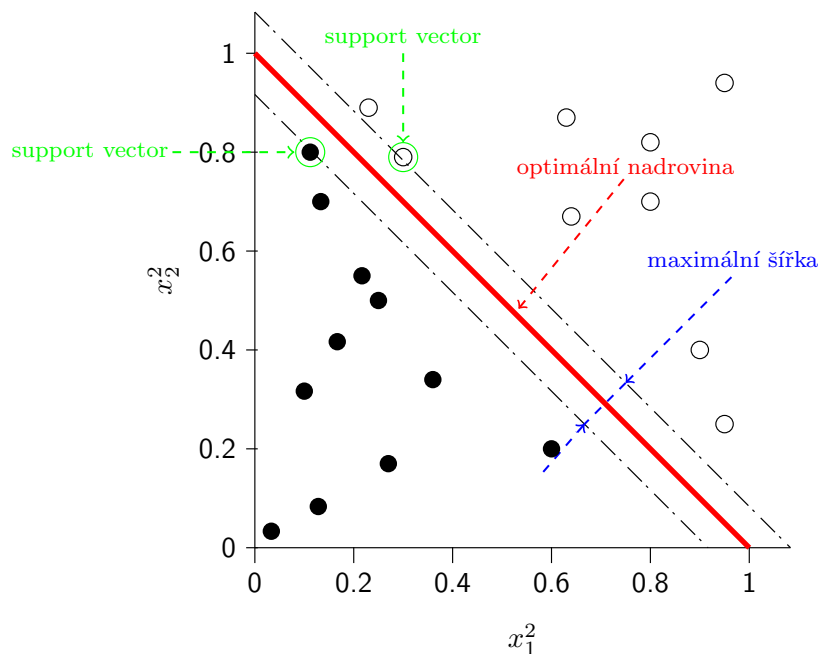
SVM Support vector machines

- První algoritmus prisuzován Vladimíru Vapnikovi (1963)
- Učební modely, které jsou velmi populární v oblasti strojového učení.
- Založena na tzv. **jádrových algoritmech** (kernel machines) s využitím **podpůrných vektorů** (support vectors).
- Původně tato technika sloužila k vytvoření optimálního binárního klasifikátoru, později byla rozšířena na řešení **problému regrese a shlukování**.
- Byly úspěšně použity ve třech hlavních oblastech: kategorizace textu, rozpoznání obrazu a bioinformatika (např. třídění novinových zpráv, rozpoznávání ručně psaných čísel nebo například vzorky rakovinových tkání).

Primárním cílem SVM je nalézt **nadrovinu**, která **optimálně rozděluje prostor** příznaků tak, aby trénovací data náležela do konkrétních tříd. Tuto nadrovinu ilustruje obrázek

17. Pokud mezera mezi oddělovací nadrovinou a nejbližšími vektory příznaků z obou kategorií (v případě binárního klasifikátoru) je maximální, jedná se o optimální řešení. Vektory příznaků v blízkosti této nadrovinu se nazývají podpůrné vektory, což znamená, že pozice ostatních vektorů nemá vliv na nadrovinu (rozhodovací funkce).

Jinými slovy, se jedná o diskriminační klasifikátor formálně definovaný rozdělovací nadrovinou, která kategorizuje nové příklady.



Obrázek 17: Optimální oddělovací hranice

Implementaci SVM lze nalézt v již existujících knihovnách, jako jsou například LIBSVM, kernlab, scikit-learn, SVMLight..

- **C –Support vektorová klasifikace (C –SVC)** – Umožňuje nedokonalé oddělení tříd pro n –tříd ($n > 2$) s postihovým multiplikátorem C , pro odlehlé hodnoty ($C > 0$).
- **ν –Support vektorová klasifikace (ν –SVC)** – n –třídní klasifikace s možností nedokonalé separace. Tato klasifikace přidává nový parametr $\nu \in (0; 1)$, čím větší je jeho hodnota, tím hladší je rozhodovací funkce.
- **Distribuční odhad (Jednotřídní SVM)** – Distribution Estimation (One-class SVM), jak již název sám o sobě napovídá všechny trénovací data pocházejí z jedné třídy, SVM vytvoří hranici, která odděluje třídu od zbývajících částí.
- **ε –Support vektorová regrese (ε –SVR)** – Vzdálenost mezi vektory příznaků a rozdělovací nadrovinou musí být menší než mez tolerance ε . Pro odlehlé hodnoty opět použijeme multiplikátor C . Musí tedy platit: $C > 0$ a $\varepsilon > 0$.
- **ν –Support vektorová regrese (ν –SVR)** – Tato klasifikace je podobná jako ε –SVR. Na místo ε se použije parametr $\nu \in (0; 1)$.

Účinnost SVM závisí na výběru správného jádra a jeho parametrů. Často se používá Gaussovo jádro s jedním parametrem γ . Díky jeho přesnosti, ale je časově náročné. V této knihovně se můžeme setkat s následujícími jádry.

- **Lineární jádro** – Použití tohoto jádra je velmi rychlé (bez jakékoliv transformace), jedná se o lineární diskriminaci a rozdělovací nadrovina bude vždy přímka. Pro toto jádro platí

$$K(x_i, x_j) = x_i^T x_j,$$

kde x_i a x_j jsou vektory vstupního prostoru.

- **Polynomické jádro** – Polynomické jádro umožňuje učení nelineárních modelů

$$K(x_i, x_j) = (\gamma x_i^T x_j + c)^d, \gamma > 0,$$

kde: $c \geq 0$, volný parametr, který vylučuje vliv vyššího řádu oproti polynomu nižšího řádu (pokud $c = 0$, jádro je homogenní), řád polynomu určuje parametr d .

- **Gaussovo jádro** – Gaussovo neboli RBF (Radial Basis Function) jádro se řadí mezi nejpoužívanější a je definované jako

$$K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}, \gamma > 0,$$

kde: $\|x_i - x_j\|^2$ značí kvadratickou euklidovskou vzdálenost mezi dvěma vektory příznaků.

- **Sigmoidní jádro** – toto jádro je podobné sigmoidní funkci v logistické regresí

$$K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r),$$

kde r je volitelný parametr.

- **Exponenciální jádro** – Exponenciální jádro χ^2 je podobné RBF jádru a využívá se převážně na histogramy

$$K(x_i, x_j) = e^{-\gamma \chi^2(x_i, x_j)}, \chi^2(x_i, x_j) = \frac{(x_i - x_j)^2}{(x_i + x_j)}, \gamma > 0,$$

- **Jádro histogramu průsečíků** – Toto jádro je také známé jako *Min Kernel*, jedná se o nejnovější jádro v této knihovně a je velmi rychlé a užitečné při klasifikaci

$$K(x_i, x_j) = \min(x_i, x_j).$$

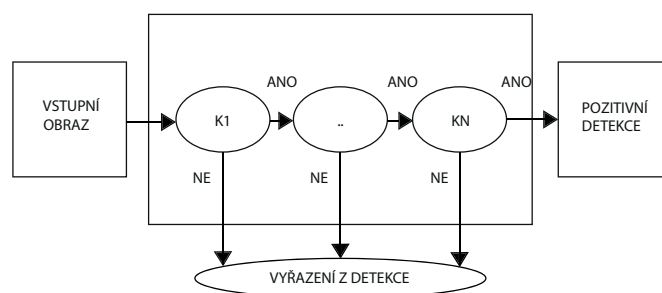
9.5.1 KNN - K-nearest neighbours

- velmi rychlý, jednoduchý
- pomocí k sousedů určíme label dotazovaného
- pokud bereme v potaz jejich vzdálenosti jedná se o modifikované KNN
- učení je velmi rychlé (jen se uloží data do struktur)
- určení se zpomaluje se zvyšujícím se k

9.5.2 Kaskádové klasifikátory

- Skládá z **více slabších** klasifikátorů umístěných v **kaskádách** za sebou.
- Požadavky na tento druh klasifikátoru byly **rychlost** detekce, aby mohl být implementován na procesorech s nižším výkonem. (v kamerách, v telefonech..)
- Klasifikátory si mezi sebou **předávají všechny** informace o vstupním obraze (může se redukovat čas, nutný pro detekci v daném obraze).
- Prvním takovým klasifikátorem byl detektor obličeje **Viola–Jones**

Klasifikátor na první vrstvě může vyfiltrovat většinu negativních oken. Na druhé vrstvě se mohou odfiltrout „těžší“ negativní okna, která přežila z první vrstvy a tak dále. Subokno, které přežije všechny vrstvy, bude označeno jako pozitivní detekce. Příklad řetězce kaskádového klasifikátoru je ilustrován na obrázku 18, kde $K1-KN$ je klasifikátor první až n -té vrstvy.



Obrázek 18: Ukázka pipeline kaskádového klasifikátoru

9.5.3 AdaBoost

- **AdaBoost**, neboli Adaptive Boosting
- klasifikátor **kombinuje** slabé klasifikátory k vytvoření jednoho silného klasifikátoru

V kombinaci více klasifikátorů s výběrem trénovací sady v každé iteraci algoritmu a přidělení správné váhy na konci trénování, docílíme klasifikátoru s dobrou přesností. Klasifikátory v tomto řetězci, které mají klasifikační přesnost menší než 50%, jsou ohodnoceny zápornou vahou. Váhou nula jsou ohodnoceny klasifikátory, které mají přesnost 50%. Pouze ty, které mají přesnost vyšší než 50%, jsou přínosné do této kombinace a můžeme hovořit o zesílení (boosting) klasifikace.

9.6 Deep learning

- Deep learning neboli **hluboké učení**, známé také jako hierarchické učení, je **sběrka algoritmů** používaných ve strojovém učení.

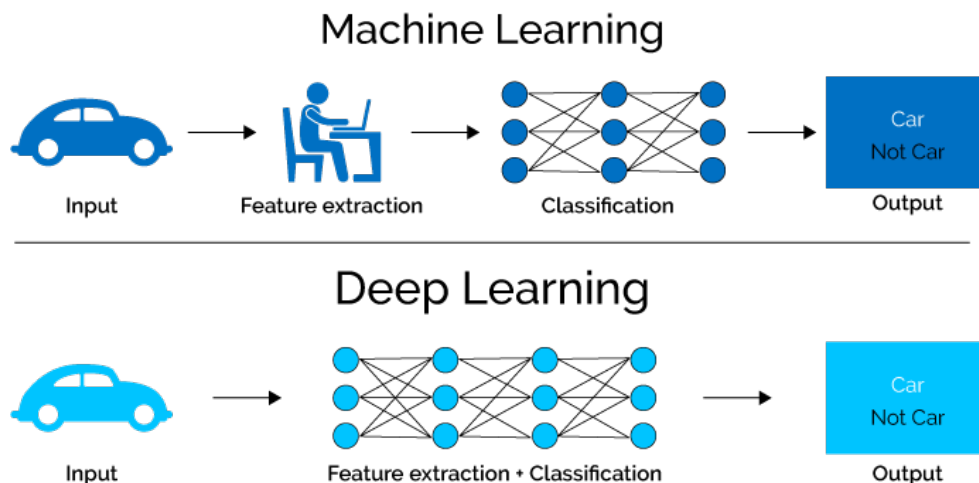
- Používají se k modelování abstrakcí na vysoké úrovni v datech za pomoci modelových architektur, které se skládají z několika nelineárních transformací.
- Hluboké učení je součástí široké skupiny metod používané pro strojové učení, které jsou založeny na učení reprezentace dat.

Hluboké strukturované učení může být:

- **Kontrolované (s učitelem)** - všechna data jsou kategorizovaná do tříd, algoritmy se učí předpovídat výstup ze vstupních dat.
- **Částečně kontrolované** - data jsou částečně kategorizovaná do tříd. Při tomto přístupu učení lze využít kombinaci kontrolovaného a nekontrolovaného přístupu učení.
- **Nekontrolované (bez učitele)** - data nejsou kategorizovaná do tříd, algoritmy se učí ze struktury vstupních dat.

Hluboké učení je specifický přístup, použitý k budování a učení neuronových sítí, které jsou považovány za velmi spolehlivé rozhodovací uzly. Jestliže vstupní data algoritmu procházejí řadou nelinearit a nelineárních transformací, tak tento algoritmus je považován za „deep“ algoritmus.

Odstraňuje také ruční identifikaci příznaků (obrázek 19) z dat a místo toho se spoléhá na jakýkoliv trénovací proces, které má za úkol zjistit užitečné vzory ve vstupních příkladech. To dělá neuronovou síť jednodušší a rychlejší, a může přinést lepší výsledky než z oblasti umělé inteligence.



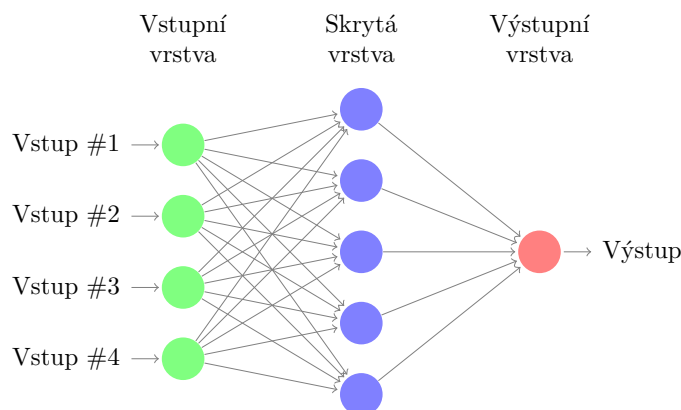
Obrázek 19: Hlavním rozdílem mezi strojovým a hlubokým učením je ten, že u strojového se příznaky musí extrahovat manuálně. [?]

9.6.1 Neuronové sítě *ANN* - Artificial Neural Network

- Inspirované **lidským mozkem**, který je složený z různých vzájemně propojených vrstev neuronů, kde každý z nich přijímá informaci z předchozího, zpracovává tuto informaci a odesílá jí do dalšího neuronu, dokud není přijat konečný výstup.

- Může se jednat o výstup s **danou kategorií**, jestliže se jedná o kontrolované učení nebo o **určitá kritéria** v případě nekontrolovaného učení.
- Umožňuje klasifikovat více tříd.

Příklad topologie neuronové sítě je na obrázku 20.



Obrázek 20: Neuronová síť je propojená skupinou uzlů, podobná síti neuronů v mozku.

Typickým příkladem neuronové sítě je **vícevrstvý perceptron** (ANN–MLP). Tato neuronová síť se skládá minimálně ze tří vrstev uzlů (vstupní, výstupní a skrytou). Každý z uzlů je **neuron**, který využívá nelineární aktivační funkci s výjimkou vstupních uzlů:

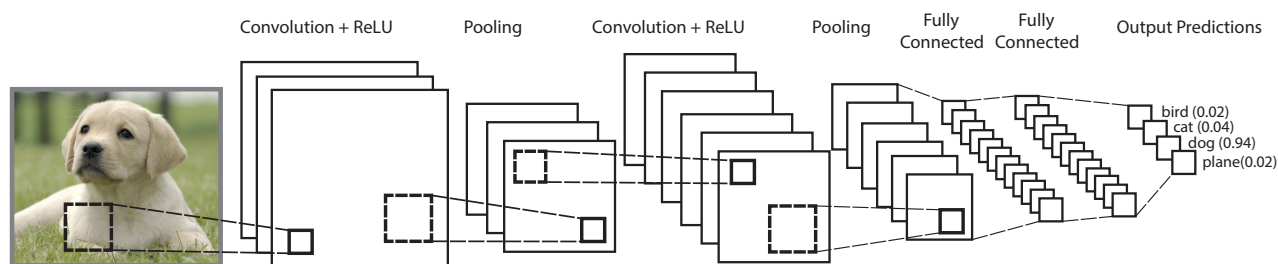
- **Vstupní vrstva** - jedná se o pasivní vrstvu, která nemodifikuje data, pouze je získává z okolního světa a pošle je dál do sítě. Počet uzlů v této vrstvě závisí na množství příznaků nebo deskriptivních informací, které chceme extrahovat z obrázku.
- **Skrytá vrstva** - v této vrstvě probíhá transformace vstupů do něčeho, co může výstupní nebo jiná skrytá vrstva využít (za předpokladu, že existuje více skrytých vrstev). Počet uzlů je určen složitostí problému a přesností, které chceme přidat do sítě.
- **Výstupní vrstva** - tato vrstva musí také vždy existovat v topologii sítě, ovšem počet uzlů v tomto případě bude definován vybranou neuronovou sítí. Pokud detekujeme na obrázku pouze jeden objekt, bude mít vrstva jen jeden uzel (lineární regrese) a bude vracet hodnotu definující pravděpodobnost konkrétního objektu v rozmezí $[-1, 1]$.
- Vysoká dimenze vstupního vektoru zvyšuje přesnost výsledků (ovšem zvyšuje výpočetní náklady)
- Aktivační funkce pro skrytou vrstvu, která umožňuje přizpůsobit nelineární hypotézy a získat lepší detekci vzoru v závislosti na poskytnutých datech (Sigmoid, tanh, ReLU).
- Hodnoty jsou získávány z předchozí vrstvy, sečteny s určitými váhami a hodnotou zkreslení. (Suma těchto hodnot je transformována pomocí aktivační funkce, může se lišit pro různé neurony)

9.6.2 Konvoluční neuronové sítě *CNN* - Convolution neural network

- Speciálním druhem vícevrstevných neuronových sítí a jsou navrženy tak, aby rozpoznaly vizuální vzory přímo z pixelu obrazu s minimálním předzpracováním.
- Mohou rozpoznat vzory s extrémní variabilitou (například ručně psané znaky) a odolnost vůči deformacím a jednoduchým geometrickým transformacím.
- Síť využívá matematickou operaci zvanou konvoluce alespoň v jedné jejích vrstvě.

Nejnámější a nejvíce používanou konvoluční neuronovou sítí jsou modely LeNet. Hlavní kroky LeNet sítě jsou:

- **Konvoluce** - tyto vrstvy provádějí konvoluci nad vstupy do neuronové sítě.
- **Nelinearita (ReLU)** - tato vrstva je použita po každé konvoluční vrstvě a jejím cílem je nahrazení všech negativních pixelů nulou ve výstupu této vrstvy (příznaková mapa).
- **Pooling/sub sampling** - ze vstupního obrazu vyextrahuje pouze zajímavé části pomocí některých matematických operací (max, avg, sum), a tím se redukuje jeho dimenzionalita.
- **Fully connected layer/klasifikace** - tato vrstva vychází z původních umělých neuronových sítí, konkrétně z vícevrstvého perceptronu. Tato vrstva je typicky umístěna na konci sítě a je propojena s klasifikační vrstvou pro predikci.



Obrázek 21: Řetězec LeNet konvoluční neuronové sítě

9.7 Bag of words BoW

BoW model může být aplikován pro klasifikaci obrázků, zachází s příznaky obrázků jako se slovy

1. Vstupem je vektor příznaků
2. centroidy z výstup k-means se stanou „slovníkem“
3. poté když máme obraz, můžeme příznaku (slovu) přiřadit třídu ze slovníku
4. vytvoříme histogram počtu výskytů slov ze slovníku
5. tento histogram dáme klasifikátoru