KMI/UMIN Umělá inteligence

Deep Learning

7. Konvoluční neuronové sítě

Jan Konečný

Katedra informatiky, PřF UP v Olomouci

9. listopadu 2026

Co nás dnes čeká

Proč ne plně propojená síť

Konvoluce, pooling, rozměry

Zpětný průchod

Od LeNetu k ResNetu

Interpretace a vizualizace

Obrazová data mají strukturu

Sousední pixely spolu souvisejí, hrana vypadá stejně v levém horním rohu jako uprostřed a tvar se nemění, když se objekt posune o několik pixelů.

Konvoluční síť tyto vlastnosti nemusí objevovat z dat: má je zabudované ve způsobu propojení svých vrstev.

Skripta: kap. 7 *Konvoluční neuronové sítě*

Tři nevýhody plného propojení

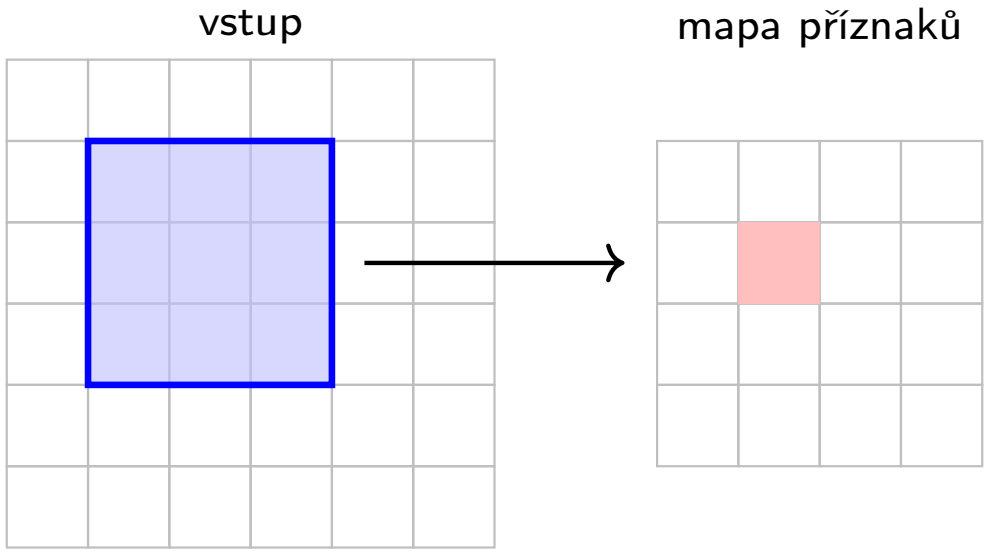
1. **Počet parametrů** roste se součinem rozměru vstupu a počtu výstupních jednotek.
2. **Chybějící lokalita:** vrstva nerozlišuje blízké a vzdálené pixely, přestože obrazové rysy vznikají z lokálních sousedství.
3. **Slabá škálovatelnost:** vyšší rozlišení přímo zvětšuje počet vah a týž lokální rys je nutné učit zvlášť na každé pozici.

Příklad 1 (Kolik vah ušetří sdílení filtru)

Obraz $224 \times 224 \times 3$ obsahuje 150 528 hodnot. Plně propojená vrstva s 1 000 neurony by měla přibližně 150,5 milionu vah. Konvoluční vrstva se 64 filtry velikosti 3×3 má naproti tomu $64\,(3 \cdot 3 \cdot 3 + 1) = 1\,792$ parametrů včetně biasů.

Filtr a mapa příznaků

Filtr (konvoluční jádro) je malá matice, například 3×3. Posouvá se po vstupu; v každé poloze se jeho prvky po složkách vynásobí s odpovídajícím výřezem a sečtou. Tak vzniká jedna hodnota výstupní mapy příznaků.



Hodnoty ve filtru se nezadávají ručně, ale učí se spolu s ostatními vahami. Filtrů se v jedné vrstvě používají desítky až stovky a každý reaguje na jiný rys: jeden na svislou hranu, druhý na určitou texturu, třetí na přechod mezi barvami.

Filtr má hloubku vstupu

Pro vstup s C_{in} kanály má každý filtr rozměr $k \times k \times C_{\text{in}}$; vrstva s C_{out} filtry má tedy

$$C_{\text{out}}\left(k^2\, C_{\text{in}} + 1\right)$$

parametrů, nezávisle na rozlišení obrazu.

Příklad 2 (Malá síť)

Vstup $32 \times 32 \times 3$, tři konvoluce 3×3 s 32, 64 a 64 filtry:

$$32(27+1) + 64(288+1) + 64(576+1) = 896 + 18\,496 + 36\,928 = 56\,320.$$

Plně propojená vrstva nad zploštěným vstupem 3 072 hodnot by přitom měla už při 512 neuronech přes 1,5 milionu vah.

Počet parametrů tedy neurčuje velikost obrazu, ale velikost jádra a počet kanálů.

Krok, doplnění okrajů, rozměr výstupu

- Krok (*stride*) určuje, o kolik pixelů se jádro posune. Větší krok zmenšuje výstup a urychluje výpočet, může však potlačit detaily.
- Doplnění okrajů (*padding*) přidává kolem vstupu hodnoty, obvykle nuly. *Valid* padding výstup zmenší, *same* padding při kroku 1 zachová rozměr.

Příklad 3 (Rozměr výstupní mapy)

Pro vstup 32×32, jádro 3×3, $P = 1$ a $S = 2$ je

$$\left\lfloor \frac{32 + 2P - 3}{S} \right\rfloor + 1 = \left\lfloor \frac{31}{2} \right\rfloor + 1 = 16.$$

Vrstva se 64 filtry tedy vytvoří tenzor $16 \times 16 \times 64$.

Zaokrouhlení dolů není detail: ne každá kombinace parametrů pokryje pravý okraj symetricky.

Pooling

Pooling redukuje rozměry mapy příznaků, a tím snižuje výpočetní náročnost dalších vrstev. Současně zvyšuje odolnost vůči malým posunům.

Max pooling

Bere největší hodnotu v okně. Odpovídá otázce „je ten rys v okně přítomen?“

Average pooling

Bere průměr hodnot v okně, tedy „jak silně?“ Uvnitř sítě se dnes používá spíše max pooling; průměrování najdeme obvykle až na konci.

Část informace se ztratí: po max poolingu už nevíme, na které pozici okna rys byl. Právě tato ztráta činí výsledek méně citlivým na drobné posuny.

Demonstrace: konvoluce krok za krokem

Vstupem je svislý světlý pruh na tmavém pozadí, filtr reaguje na svislou hranu:

$$\boldsymbol{X} = \begin{pmatrix} 0 & 3 & 3 & 3 & 0 \\ 0 & 3 & 3 & 3 & 0 \\ 0 & 3 & 3 & 3 & 0 \\ 0 & 3 & 3 & 3 & 0 \\ 0 & 3 & 3 & 3 & 0 \end{pmatrix}, \qquad \boldsymbol{W} = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$

poloha 1

0	3	3	3	0
0	3	3	3	0
0	3	3	3	0
0	3	3	3	0
0	3	3	3	0

$y_{11} = 3 \cdot (0 - 3) = -9$

poloha 2

0	3	3	3	0
0	3	3	3	0
0	3	3	3	0
0	3	3	3	0
0	3	3	3	0

$y_{12} = 3 \cdot (3 - 3) = 0$

poloha 3

0	3	3	3	0
0	3	3	3	0
0	3	3	3	0
0	3	3	3	0
0	3	3	3	0

$y_{13} = 3 \cdot (3 - 0) = 9$

Levá hrana dá zápornou odezvu, vnitřek pruhu nulovou a pravá hrana kladnou.

Demonstrace: ReLU, pooling a zpětný průchod

Celá výstupní mapa je (řádky se opakují, protože vstup je po výšce konstantní)

$$\boldsymbol{Y} = \begin{pmatrix} -9 & 0 & 9 \\ -9 & 0 & 9 \\ -9 & 0 & 9 \end{pmatrix} \longmapsto \mathrm{ReLU}(\boldsymbol{Y}) = \begin{pmatrix} 0 & 0 & 9 \\ 0 & 0 & 9 \\ 0 & 0 & 9 \end{pmatrix} \longmapsto \begin{pmatrix} 0 & 9 \\ 0 & 9 \end{pmatrix}$$

- ReLU zahodí zápornou odezvu, takže filtr hlásí jen přechod *světlá* $\rightarrow$ *tmavá* (zleva doprava). Opačnou hranu by musel detekovat jiný filtr.
- Pooling s oknem 2×2 a krokem 1 zmenší mapu na 2×2 a ponechá informaci „v pravé části okna hrana je“.

Zpětný průchod nad tímto příkladem

Přijde-li do levé horní buňky výsledku gradient δ, projde poolingem na pozici maxima. Tam je ale výstup ReLU nulový, takže se dál nedostane. Konvoluční váhy dostanou nenulový gradient jen z pozic, kde filtr skutečně vydal kladnou odezvu.

Ekvivariance a invariance

Konvoluce je ekvivariantní vůči posunu

Označme T_δ posun obrazu o vektor $\boldsymbol{\delta}$. Pak

$$(T_\delta \boldsymbol{x}) * \boldsymbol{w} = T_\delta(\boldsymbol{x} * \boldsymbol{w}),$$

tedy posuneme-li vstup, posune se mapa příznaků o totéž. Odezva se nemění, mění se jen její poloha.

Invariance je něco jiného: výsledek se nezmění vůbec. Té se v síti dosahuje až agregací — poolingem a zejména globálním průměrováním na konci, kdy se prostorová poloha zahodí.

Co zabudované není

Otočení, změna měřítka ani změna osvětlení z konvoluce neplynou. Odolnost vůči nim se získává augmentací dat, nebo se do architektury musí zavést zvlášť.

Co se filtry naučí

První vrstva se obvykle naučí detektory hran a barevných přechodů. Hlubší vrstvy kombinují jejich výstupy do složitějších tvarů.



Tato interpretace je užitečná, ale nelze ji brát doslova: naučená informace bývá rozložena mezi více filtrů a jednotlivý filtr nemusí odpovídat lidsky pojmenovatelnému rysu.

Pooling, nebo krokovaná konvoluce?

Rozlišení lze snížit dvěma způsoby:

- **pooling** — pevná operace bez parametrů,
- **konvoluce se stridem** > 1 — naučená redukce.

Novější architektury často pooling nahrazují krokovanou konvolucí: síť si sama určí, co při zmenšení zachovat. Cenou jsou další parametry a o něco vyšší riziko přeučení.

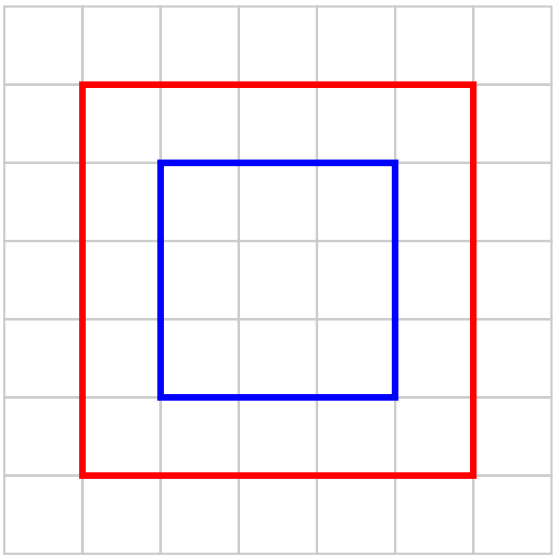
Na konci sítě

Místo zploštění a velké plně propojené vrstvy se dnes obvykle používá *globální průměrování* přes prostorové rozměry: z mapy $H \times W \times C$ vznikne vektor délky C. Ušetří to většinu parametrů klasifikátoru.

Receptivní pole

Příklad 4 (Jak roste s hloubkou)

Jeden neuron po první konvoluci 3×3 vidí oblast 3×3 vstupu. Druhá konvoluce 3×3 se stridem 1 zvětší receptivní pole na 5×5, nikoli na 6×6, protože se sousední okna překrývají.



po první konvoluci: 3×3

po druhé konvoluci: 5×5

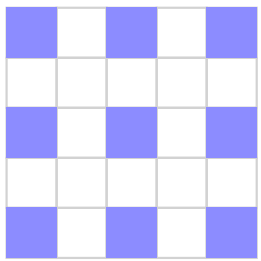
Hloubka tedy dovoluje skládat lokální hrany do stále větších tvarů, aniž by bylo nutné použít velké jádro.

Dilatovaná konvoluce

Jádro se nepokládá na souvislé okno, ale na mřížku s rozestupem r:

$$y(i) = \sum_a w(a)\, x(i + r\, a).$$

Efektivní velikost jádra je $k + (k-1)(r-1)$, počet parametrů se nemění.



$r = 2$: jádro 3×3 pokrývá oblast 5×5

Tři vrstvy 3×3 s $r = 1, 2, 4$ dají receptivní pole 15×15 $(1 + 2 + 4 + 8)$ při nezměněném počtu vah a bez zmenšení rozlišení — toho se využívá u segmentace a u generování zvuku.

F. Yu, V. Koltun: Multi-Scale Context Aggregation by Dilated Convolutions. ICLR (2016). — A. van den Oord a kol.: WaveNet (2016).

Separabilní konvoluce

Běžnou konvoluci lze rozložit na dva levnější kroky:

1. depthwise — jádro $k \times k$ zvlášť pro každý kanál,
2. pointwise — konvoluce 1×1, která kanály promíchá.

Příklad 5 (Kolik se ušetří)

Pro $C_{\text{in}} = C_{\text{out}} = 256$ a $k = 3$ má běžná konvoluce $9 \cdot 256^2 = 589\,824$ vah, rozklad jen $9 \cdot 256 + 256^2 = 67\,840$, tedy zhruba devětkrát méně. Obecně je poměr

$$\frac{k^2 C + C^2}{k^2 C^2} = \frac{1}{C} + \frac{1}{k^2}.$$

Prostorové a kanálové mísení se tím oddělí. Za nižší cenu se platí menší vyjadřovací silou jedné vrstvy; architektury to kompenzují větším počtem vrstev.

A. G. Howard a kol.: MobileNets (2017). — F. Chollet: Xception: Deep Learning with Depthwise Separable Convolutions. CVPR (2017).

Konvoluce jako síť se sdílenými vahami

Pro jednorozměrný vstup $x_1, \ldots, x_4$ a filtr (w_1, w_2) s krokem 1 počítá dopředný průchod

$$y_1 = w_1 x_1 + w_2 x_2, \quad y_2 = w_1 x_2 + w_2 x_3, \quad y_3 = w_1 x_3 + w_2 x_4.$$

Při zpětném průchodu přichází ke každému y_i hodnota $\delta_i = \partial E / \partial y_i$. Pro vstup x_2 se sčítají obě cesty:

$$\frac{\partial E}{\partial x_2} = w_2 \delta_1 + w_1 \delta_2.$$

Sdílené váhy

Chybová funkce závisí na w_1 přes všechny výstupy, ve kterých se použila; gradienty z jednotlivých pozic se proto **sčítají**: $\partial E / \partial w_1 = \delta_1 x_1 + \delta_2 x_2 + \delta_3 x_3$.

Zpětný průchod poolingem

Max pooling: směrovač

Derivace maxima podle vítězného vstupu je 1, podle ostatních 0. Příchozí gradient se pošle na pozici, která nesla maximum; ostatní dostanou nulu. Síť si proto musí index maxima uložit.

Average pooling: rozprostření

Derivace průměru podle každého vstupu je $1/k$, takže se příchozí gradient rozdělí rovnoměrně. U okna 2×2 obdrží každá pozice $\delta/4$.

Pooling nemá žádné váhy, takže se sám nic neučí; jediným úkolem je správně protlačit gradient do předchozí vrstvy.

Vzorce pro gradient konvoluce mají opět konvoluční strukturu: gradient vstupu odpovídá konvoluci s filtrem otočeným o 180°. Zpětný průchod tak používá stejné efektivní operace jako průchod dopředný.

Konvoluce jako řídká vrstva s vázanými vahami

Předchozí příklad lze zapsat maticově. Pro vstup délky 5 a filtr (w_1, w_2, w_3) je

$$\boldsymbol{y} = \boldsymbol{M}\boldsymbol{x}, \qquad \boldsymbol{M} = \begin{pmatrix} w_1 & w_2 & w_3 & 0 & 0 \\ 0 & w_1 & w_2 & w_3 & 0 \\ 0 & 0 & w_1 & w_2 & w_3 \end{pmatrix}.$$

Plně propojená vrstva 3×5 by měla 15 volných čísel, tato matice jen tři. Konvoluce tedy není nová operace, ale plně propojená vrstva se dvěma omezeními:

- řídkost — většina vah je pevně nulová (lokalita),
- vázání vah — zbylé váhy se opakují (sdílení mezi pozicemi).

Obojí je *induktivní preference*: informace o úloze vložená do architektury místo do dat. Proto konvoluční síť potřebuje na obrazová data podstatně méně vzorů než plně propojená.

LeNet-5 (1998)

Určen pro rozpoznávání ručně psaných znaků; na MNIST dosahoval chyby přibližně 0,8 %.

Model střídá konvoluční vrstvu a průměrný pooling, poté se mapy zploští a zpracují plně propojenými vrstvami.

Z LeNetu zůstalo dodnes:

- seskupování konvoluce a poolingu do bloků a jejich opakování,
- růst počtu filtrů s hloubkou při klesajícím rozlišení,
- oddělení *extraktoru příznaků* od *klasifikátoru*.

V přednášce o přenosu učení uvidíme, že právě v tomto místě lze síť rozdělit a extraktor použít na jinou úlohu.

Y. LeCun, L. Bottou, Y. Bengio, P. Haffner: Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE* 86(11) (1998).

AlexNet (2012)

Osm vrstev (pět konvolučních, tři plně propojené), vstup 227×227, první jádro 11×11, přibližně 60 milionů parametrů.

Hlavní inovace:

- ReLU místo sigmoidy — výrazně rychlejší trénování,
- dropout v plně propojených vrstvách,
- augmentace dat (náhodné oříznutí, zrcadlení),
- trénování na GPU nad rozsáhlým datasetem ImageNet.

Jeho úspěch neznamenal vznik hlubokého učení v tom roce; přesvědčivě však ukázal, že škálování dat a výpočtu mění dosažitelnou kvalitu modelu.

A. Krizhevsky, I. Sutskever, G. E. Hinton: ImageNet Classification with Deep Convolutional Neural Networks. NIPS (2012).

VGG: hloubka z malých jader

VGG používá 16–19 vrstev postavených převážně z konvolucí 3×3.

Proč malá jádra

Dvě konvoluce 3×3 mají stejné receptivní pole jako jedna 5×5, ale méně parametrů: $2 \cdot 9C^2 = 18C^2$ proti $25C^2$. Navíc mezi nimi leží další nelinearita.

Cenou je vysoký počet parametrů v závěrečných plně propojených vrstvách a velká paměťová náročnost aktivací. Další zvyšování hloubky už samo o sobě nepomáhalo — a to vedlo k otázce, proč.

GoogLeNet a konvoluce 1×1

Vhodná velikost filtru závisí na měřítku hledaného rysu. Inception modul proto zpracuje tentýž vstup paralelně konvolucemi 1×1, 3×3 a 5×5 a max poolingem; výstupy větví se zřetězí v kanálové dimenzi.

Konvoluce 1×1 nemění prostorové okolí, ale v každém pixelu počítá lineární kombinaci všech kanálů. Před dražší konvolucí tak může snížit například 256 kanálů na 64 a vytvořit úzké hrdlo.

Výsledek

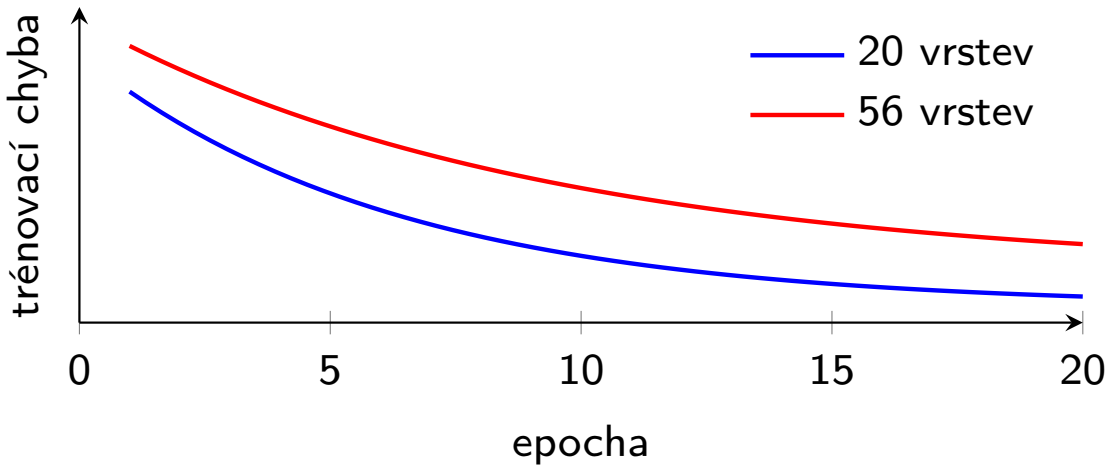
GoogLeNet měla při 22 vrstvách přibližně 6,8 milionu parametrů, zatímco AlexNet přibližně 60 milionů.

C. Szegedy a kol.: Going Deeper with Convolutions. CVPR (2015).

Problém degradace

Přidané vrstvy by teoreticky mohly realizovat identitu, hlubší běžná síť ji však nemusí gradientním učením najít.

He a kol. naměřili u 56vrstvé sítě *vyšší* chybu než u 20vrstvé, a to i na trénovacích datech. Nešlo tedy o přeučení, ale o zhoršenou optimalizaci.



Reziduální blok

Blok počítá místo celé transformace $H(\boldsymbol{x})$ jen její odchylku od identity, $F(\boldsymbol{x}) = H(\boldsymbol{x}) - \boldsymbol{x}$, takže

$$\boldsymbol{y} = F(\boldsymbol{x}) + \boldsymbol{x}.$$

Identitní zkratka nepřidává žádné parametry ani výpočty.

Konstrukce pomáhá ze dvou vzájemně se doplňujících důvodů:

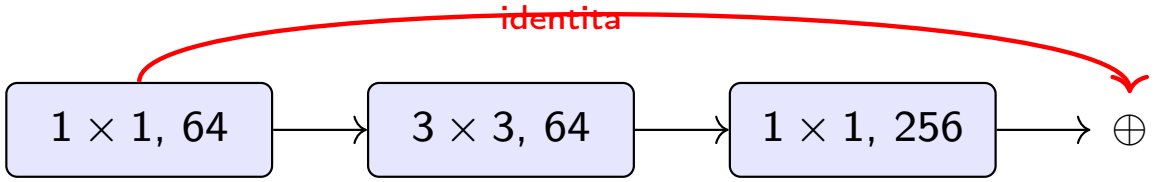
- **Snazší úloha:** je-li optimální transformace blízká identitě, stačí vytlačit váhy k nule a naučit $F \approx 0$.
- **Cesta pro gradient:** $\partial \boldsymbol{y}/\partial \boldsymbol{x} = \partial F/\partial \boldsymbol{x} + 1$, takže i kdyby se gradient uvnitř F rozpadl, jednička od identity zaručí, že do nižších vrstev doteče nezeslabený signál.

Soubor reziduálních sítí s až 152 vrstvami vyhrál ILSVRC 2015 s chybou 3,57 % (top-5). Zkratky se od té doby staly součástí prakticky všech hlubokých architektur.

K. He, X. Zhang, S. Ren, J. Sun: Deep Residual Learning for Image Recognition. CVPR (2016).

Bottleneck blok

Hluboké varianty ResNetu (50, 101, 152) nepoužívají dvě konvoluce 3×3, ale trojici $1 \times 1 \to 3 \times 3 \to 1 \times 1$: první konvoluce počet kanálů sníží, prostřední pracuje v úzkém místě a třetí kanály zase rozšíří.



Příklad 6 (Cena bloku při 256 kanálech)

Dvě konvoluce 3×3: $2 \cdot 9 \cdot 256^2 = 1\,179\,648$ vah. Bottleneck: $256 \cdot 64 + 9 \cdot 64^2 + 64 \cdot 256 = 69\,632$ vah, tedy zhruba sedmnáctkrát méně při stejné šířce vstupu i výstupu.

Pozdější varianta přesouvá normalizaci a aktivaci před konvoluce (*pre-activation*), takže zkratka vede nepřerušeně celým blokem.

K. He a kol.: Identity Mappings in Deep Residual Networks. ECCV (2016).

Přehled architektur

model	rok	hloubka	klíčový nápad
LeNet-5	1998	7	střídání konvoluce a poolingu
AlexNet	2012	8	ReLU, dropout, GPU, ImageNet
VGG	2014	16–19	hloubka z jader 3×3
GoogLeNet	2014	22	inception modul, hrdla 1×1
ResNet	2015	50–152	reziduální spojení

Pokrok nebyl jen v hloubce: každý krok řešil konkrétní překážku — saturaci aktivací, počet parametrů, cenu výpočtu, degradaci při hloubce.

Konvoluce zůstávají účinnou induktivní preferencí i v době Transformerů; v osmé přednášce uvidíme, kde se obě rodiny potkávají.

Parametry, FLOPy a paměť

model	parametry	operace na obraz 224^2
AlexNet	60 M	0,7 GFLOP
VGG-16	138 M	15,5 GFLOP
GoogLeNet	6,8 M	1,5 GFLOP
ResNet-50	25,6 M	4,1 GFLOP

Tři veličiny se nepohybují spolu:

- **Parametry** sedí hlavně v plně propojených vrstvách a v konvolucích s mnoha kanály; rozhodují o velikosti souboru s modelem.
- **Operace** vznikají hlavně v raných vrstvách, kde je rozlišení vysoké; rozhodují o době odezvy.
- **Paměť při trénování** určují uložené aktivace, ne váhy. Proto ji zvětšuje velikost dávky, i když se model nemění.

VGG je toho dobrým příkladem: proti ResNetu-50 má pětkrát více vah a zároveň skoro čtyřikrát více operací, a přesto klasifikuje hůř.

Co přišlo po ResNetu

- **DenseNet** — výstupy vrstev se místo sčítání zřetězují, takže každá vrstva vidí všechny předchozí mapy. Silná znovupoužitelnost příznaků za cenu paměti.
- **ResNeXt** — konvoluce ve skupinách; vedle hloubky a šířky přibývá třetí rozměr architektury, počet paralelních větví.
- **SENet** — blok, který se naučí kanály přeškálovat podle globálního kontextu. Jde o attention přes kanálovou dimenzi.
- **EfficientNet** — hloubka, šířka a rozlišení se zvětšují společně podle jednoho koeficientu, ne každé zvlášť.
- **ConvNeXt** — čistě konvoluční síť s postupy převzatými z Transformerů; ukazuje, že velká část jejich náskoku byla v receptu trénování, ne v architektuře.

Huang a kol.: Densely Connected Convolutional Networks. CVPR (2017). — Xie a kol.: Aggregated Residual Transformations. CVPR (2017). — Hu a kol.: Squeeze-and-Excitation Networks. CVPR (2018). — Tan, Le: EfficientNet. ICML (2019). — Liu a kol.: A ConvNet for the 2020s. CVPR (2022).

Model může mít pravdu ze špatného důvodu

Model může mít vysokou přesnost, a přesto se rozhodovat podle jiného rysu, než předpokládáme. Z testovacích čísel se to nepozná; projeví se to až v provozu, kde příslušná korelace přestane platit.

Clever Hans

Kůň Hans byl na počátku 20. století předváděn jako zvíře schopné řešit početní úlohy. Uspěl ovšem jen tehdy, když viděl tazatele a tazatel znal odpověď: reagoval na drobné změny jeho postoje, ne na obsah úlohy.

Typické příklady z hlubokého učení: tanky fotografované ve sněhu, vlci na sněhu proti psům na trávě, kůň rozpoznávaný podle fotografovy vodoznačky.

O. Pfungst: *Clever Hans*. Henry Holt (1911). — S. Lapuschkin a kol.: Unmasking Clever Hans Predictors and Assessing What Machines Really Learn. *Nature Communications* 10 (2019).

Saliency mapa

Měří citlivost skóre třídy na jednotlivé vstupní pixely:

$$\boldsymbol{M} = \left| \frac{\partial S_c(\boldsymbol{X})}{\partial \boldsymbol{X}} \right|.$$

Výpočet je drobnou variací zpětné propagace — místo derivace ztráty podle vah počítáme derivaci skóre podle vstupu:

1. dopředný průchod, spočteme $S_c(\boldsymbol{X})$;
2. na výstup třídy c zavedeme 1, ostatní 0;
3. gradient šíříme až k pixelům;
4. absolutní hodnotu zobrazíme jako teplotní mapu.

Předností je rychlost (jeden dopředný a jeden zpětný průchod). Gradient po pixelech však bývá šumový a nezachycuje abstraktní rysy vyšších vrstev.

K. Simonyan, A. Vedaldi, A. Zisserman: Deep Inside Convolutional Networks. ICLR Workshop (2014).

Grad-CAM

Místo citlivosti na jednotlivý pixel váží *mapy* poslední konvoluční vrstvy podle jejich důležitosti pro cílovou třídu:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial S_c}{\partial A_{i,j}^k}, \qquad L^c = \mathrm{ReLU}\Big(\sum_k \alpha_k^c A^k \Big).$$

ReLU zajistí, že vizualizujeme jen pozitivní vlivy. Heatmapa má rozlišení poslední konvoluční vrstvy, takže se před překrytím zvětší interpolací.

Příklad 7 (Správná predikce ze špatného důvodu)

Klasifikátor lodí dosahuje vysoké accuracy, ale Grad-CAM zvýrazňuje především vodu a oblohu. Na fotografiích lodí vytažených na břeh pak selže.

R. R. Selvaraju a kol.: Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. ICCV (2017).

Heatmapa není důkaz

Sama heatmapa nic nedokazuje, jde jen o podezření. Ověřit je lze pokusem:

- vystřihnout objekt z pozadí a změřit přesnost,
- naopak nechat pozadí a objekt zakrýt,
- nasadit model na data, kde daná korelace neplatí.

Zůstane-li přesnost i po odstranění objektu vysoká, model se skutečně dívá na pozadí.

Proč to patří do přednášky o CNN

Interpretační metody nejsou doplněk pro publikace. Jsou to nástroje ladění: odhalí zkratky, chybné anotace i únik informace, které se z agregované metriky nepoznají.