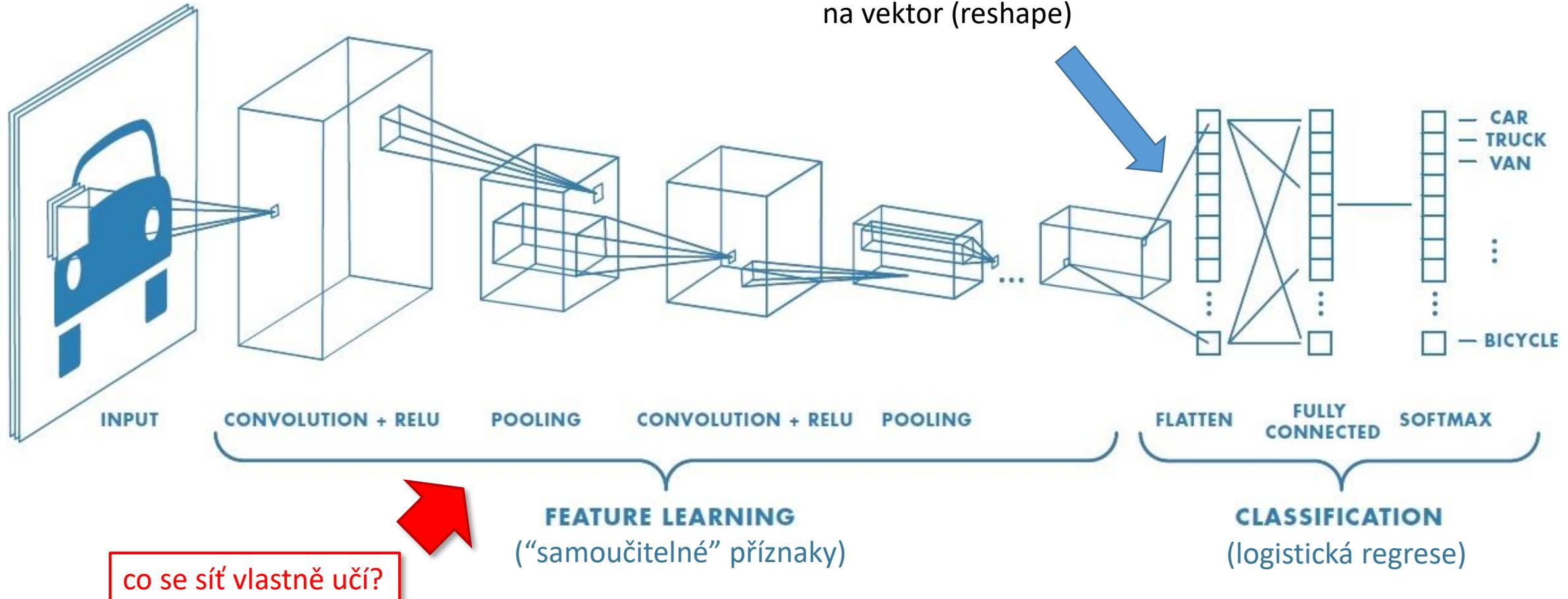


# Aplikace neutronových sítí

---

Vizualizace a analýza konvolučních sítí

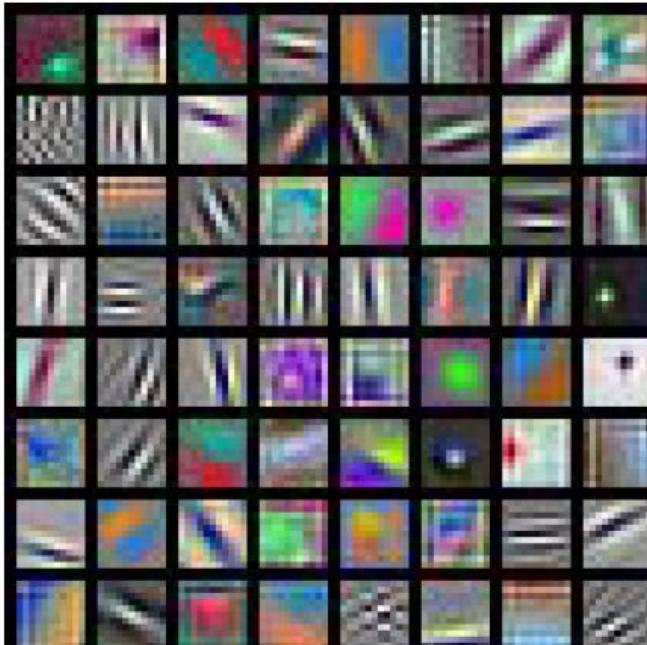
# Konvoluční síť (Convolutional Neural Network, CNN)



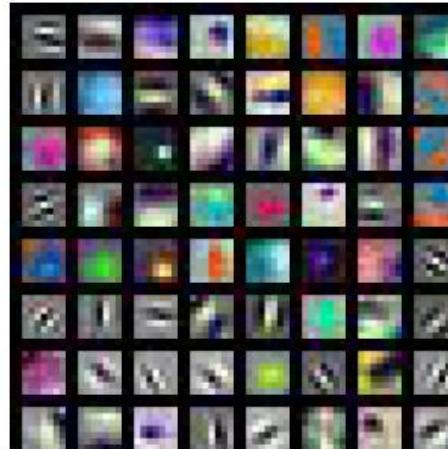
obrázek: <https://ch.mathworks.com/fr/discovery/convolutional-neural-network.html>

# Vizualizace filtrů první vrstvy

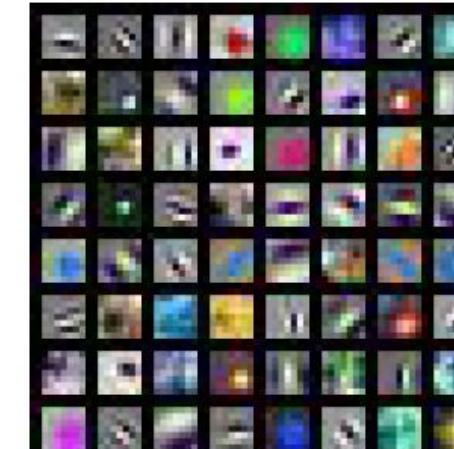
- konvoluce  $\approx$  korelace
- největší hodnotu korelace má pokud vstup a filtr jsou shodné
- váhy filtru tedy odpovídají tomu, na co nejsilněji reagují



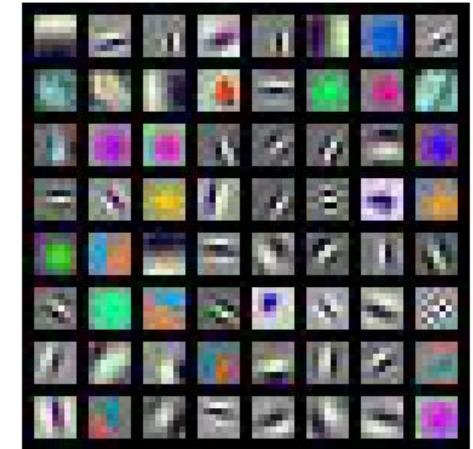
AlexNet:  
 $64 \times 3 \times 11 \times 11$



ResNet-18:  
 $64 \times 3 \times 7 \times 7$



ResNet-101:  
 $64 \times 3 \times 7 \times 7$



DenseNet-121:  
 $64 \times 3 \times 7 \times 7$

konvoluční filtry **první vrstvy** sítě

# Druhá a další vrstvy

- U dalších vrstev obtížnější, protože filtry mají např. 64 kanálů
- Jak zobrazit?
- Navíc vstupem již není obraz, ale příznakové mapy

Příklad: demo convnetjs by Karpathy; 16 filtrů 5x5x16

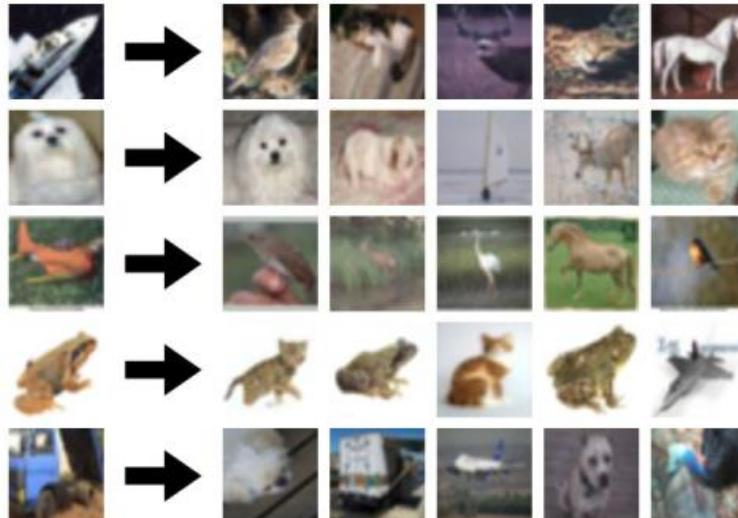
Weights:



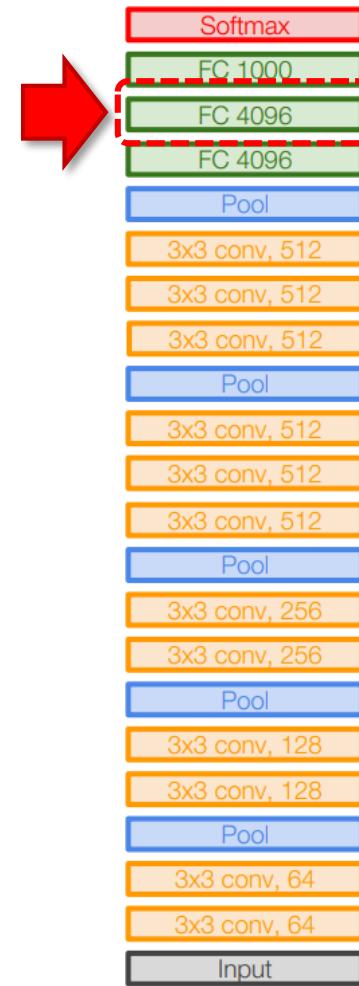
demo: <https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html>

# Vizualizace metodou nejbližšího souseda

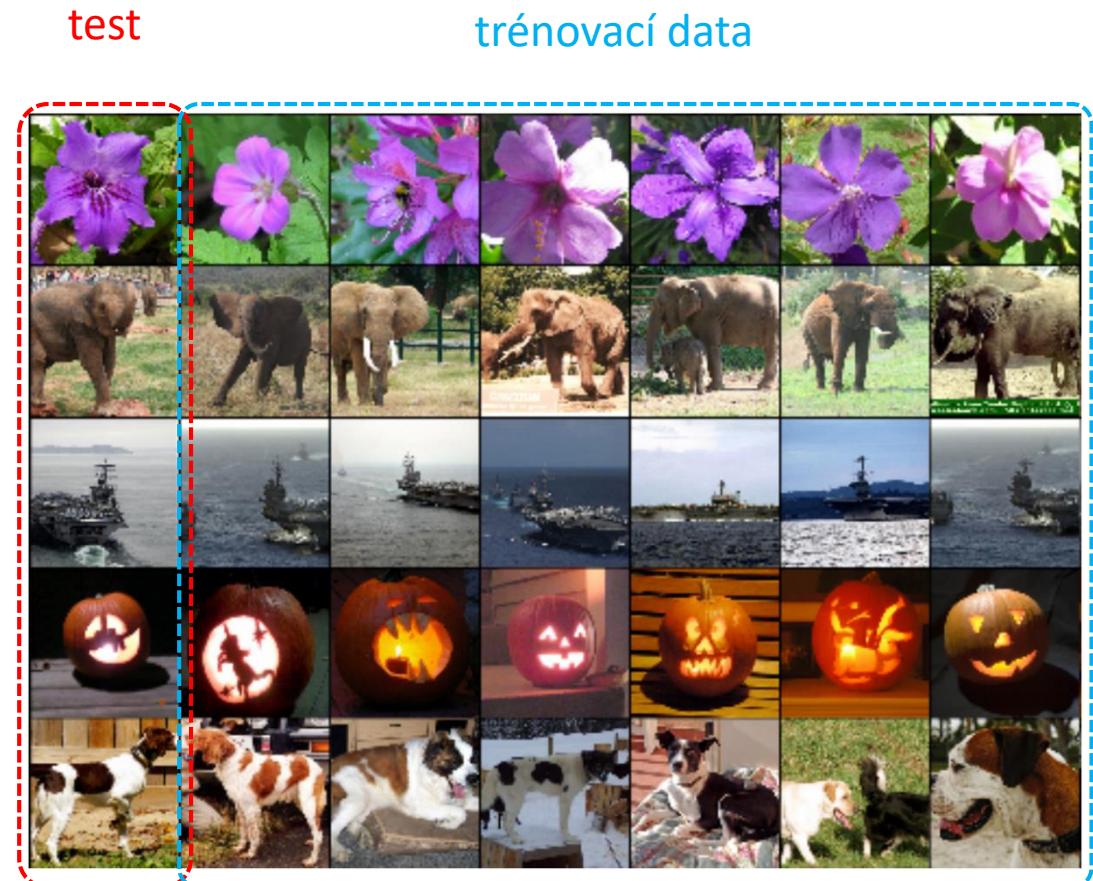
nejbližší soused dle pixelů



obrázek: <http://cs231n.stanford.edu/>



nejbližší soused dle příznakového popisu sítí



obrázek: Krizhevsky et al: ImageNet  
Classification with Deep Convolutional Neural  
Networks

# Vizualizace clusteringem (např. t-SNE)



1000x1000 (240KB)



4000x4000 (3.2MB)

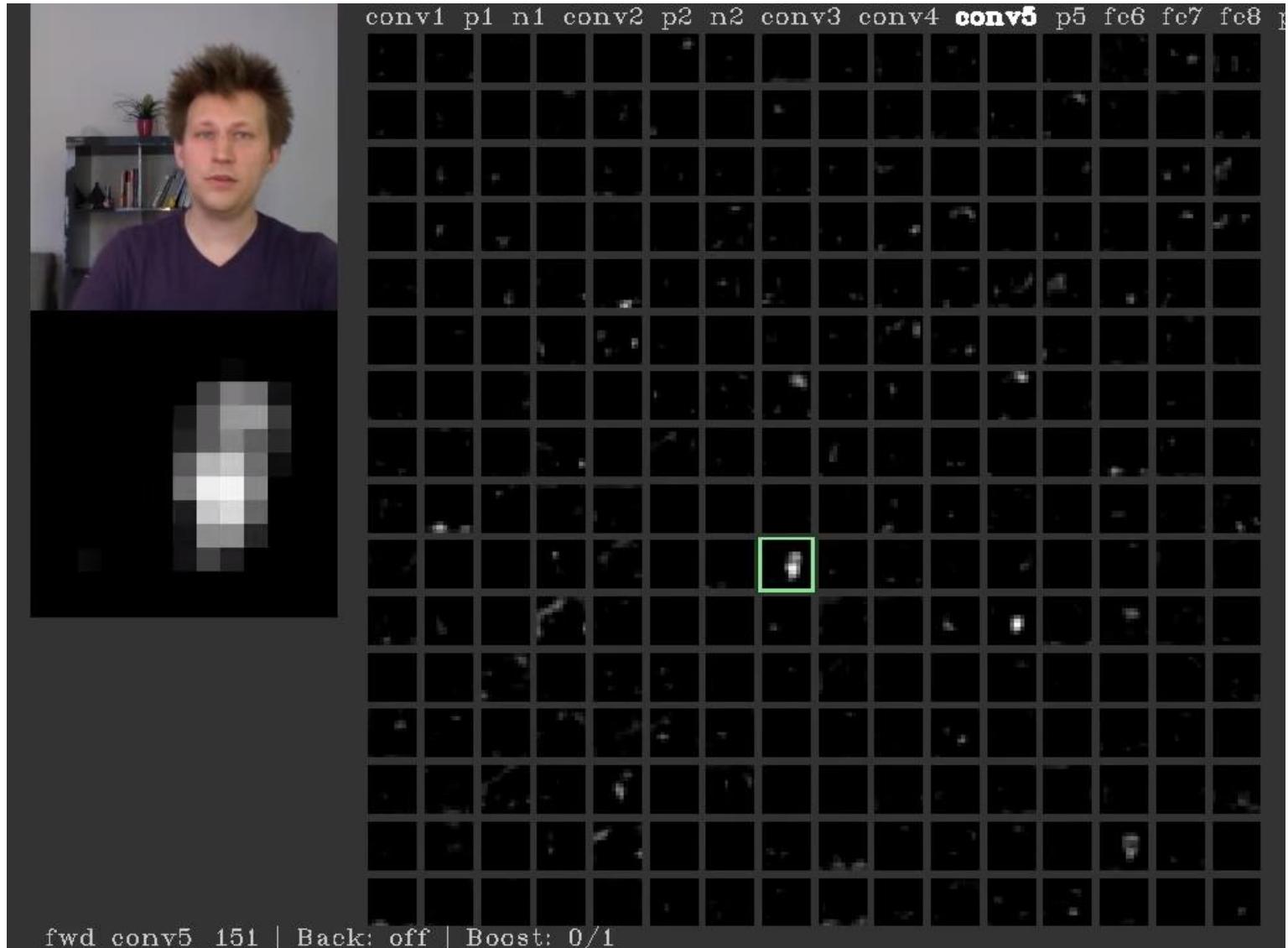


6000x6000 (6.6MB)

demo: <https://cs.stanford.edu/people/karpathy/cnnembed/>

# Zobrazení aktivací

- Namísto vah filtrů se můžeme podívat, jak se jednotlivé filtry aktivují
- Vpravo např. ukázka detektoru obličejů



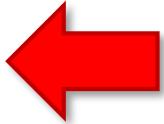
demo: <http://yosinski.com/deepvis>

# Které textury maximálně aktivují filtry?

Softmax
FC 1000
FC 4096
FC 4096
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 256
3x3 conv, 256
Pool
3x3 conv, 128
3x3 conv, 128
Pool
3x3 conv, 64
3x3 conv, 64
Input

1. vybereme vrstvu
2. projedeme spoustu obrázků
3. zapamatujeme si max. hodnoty v konvoluční mapě a kterým okénkům v obrázcích odpovídají

výsledky pro vrstvu 6

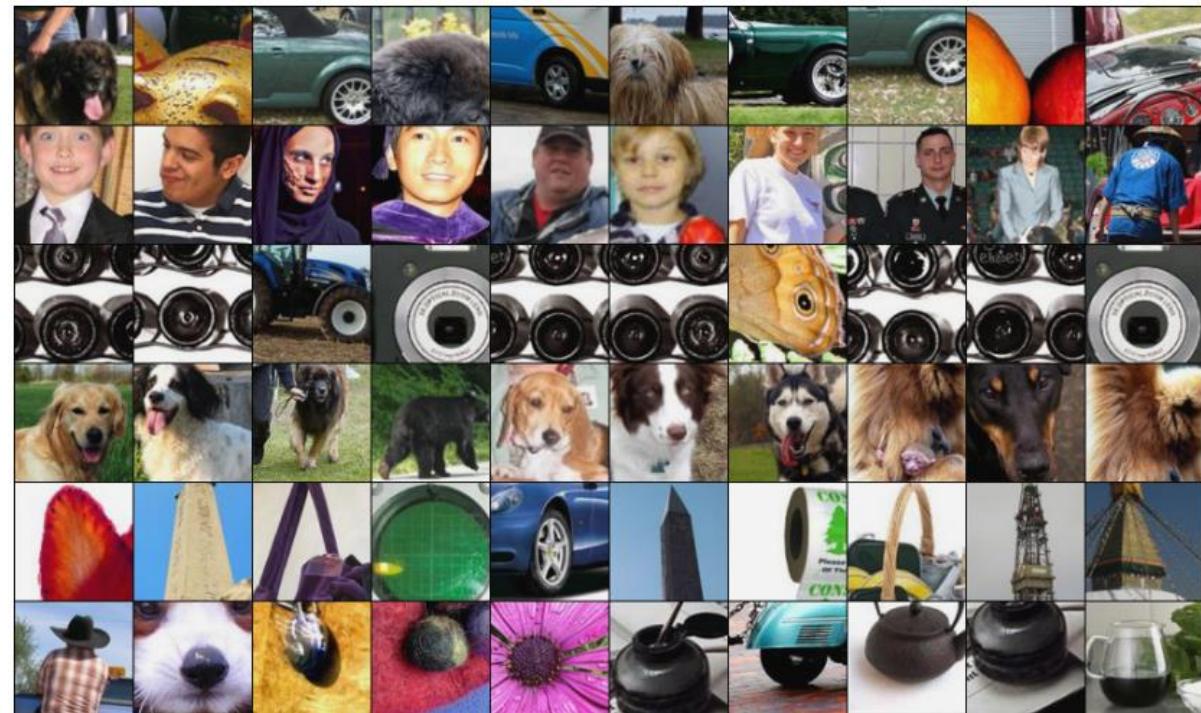


# Které textury maximálně aktivují filtry?

Softmax
FC 1000
FC 4096
FC 4096
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 256
3x3 conv, 256
Pool
3x3 conv, 128
3x3 conv, 128
Pool
3x3 conv, 64
3x3 conv, 64
Input

1. vybereme vrstvu
2. projedeme spoustu obrázků
3. zapamatujeme si max. hodnoty v konvoluční mapě a kterým okénkům v obrázcích odpovídají

výsledky pro vrstvu **9**



článek: [Springenberg et al.: Striving for Simplicity: The All Convolutional Net](#)

# Změna klasifikace v závislosti na zakrytí obrázku

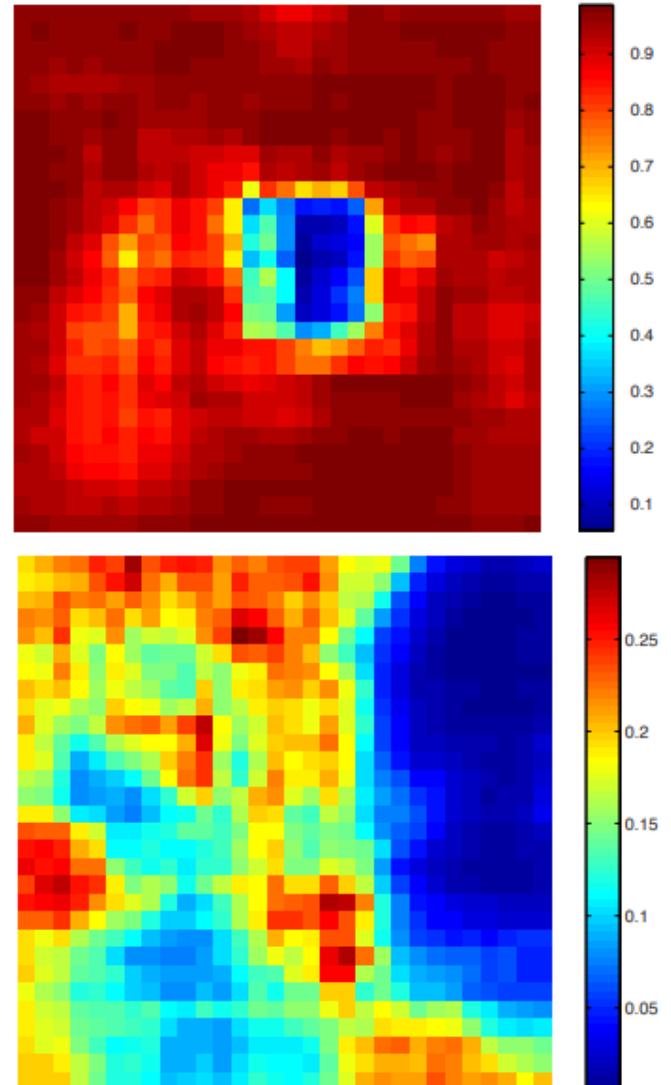
1. po obrázku posouváme šedivý čtverec, kterým zakrýváme odpovídající oblast
2. sledujeme, jak se pro každou pozici mění pravděpodobnost správné třídy
3. vykreslíme



True Label: Pomeranian



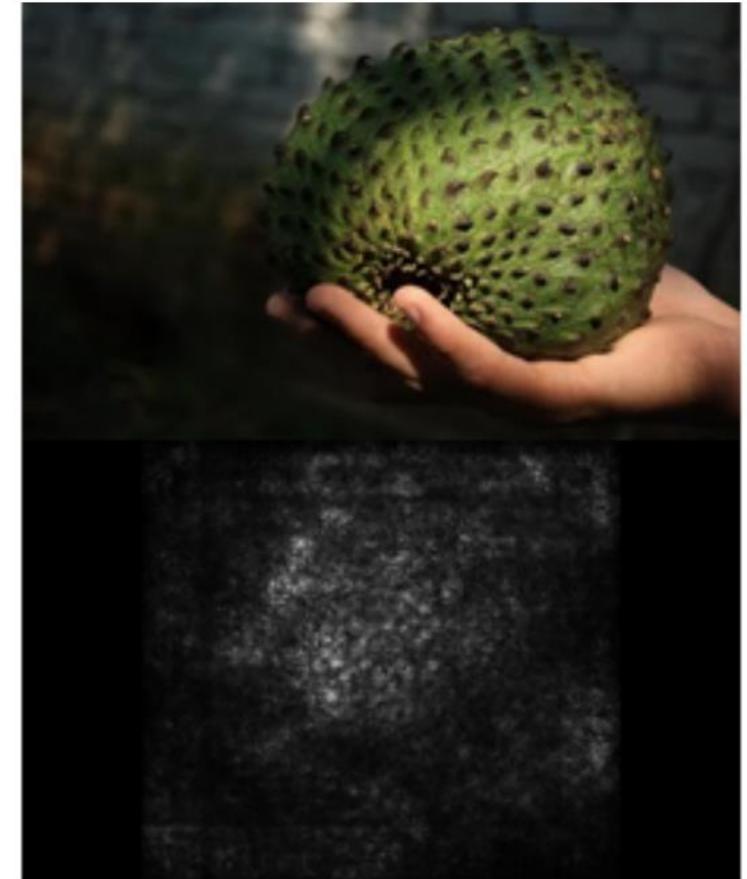
True Label: Car Wheel



článek: [Zeiler, Fergus: Visualizing and Understanding Convolutional Networks](#)

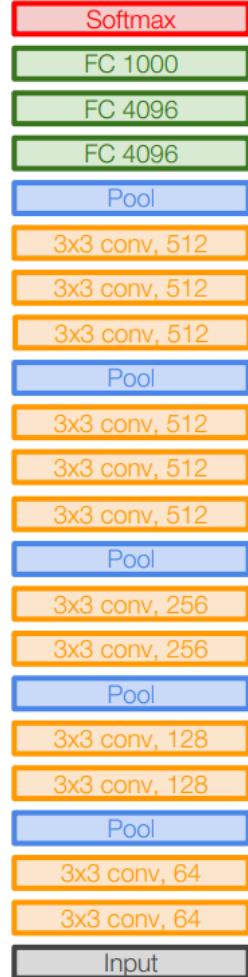
# Významnost pixelů skrze gradient

1. obrázek projedeme sítí
2. nastavíme gradient klasifikované třídy na jedna, ostatní na nuly
3. provedeme zpětný průchod –  
**nezajímají nás ovšem gradienty na váhy filtrů, ale na vstup**
4. gradient je RGB → vezmeme max. přes kanály + absolutní hodnotu
5. získáme důležitost jednotlivých pixelů pro klasifikaci (angl. saliency map)



článek: [Simonyan: Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps](#)

# Guided backprop



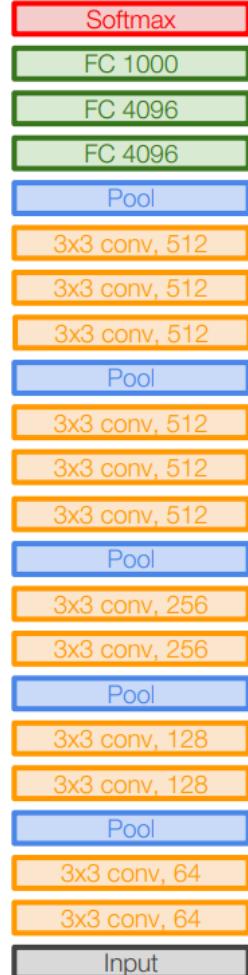
velmi podobné jako saliency mapy, ovšem s několika rozdíly:

1. začneme backprop ne na vrchu sítě, ale na zvoleném neuron nějaké vrstvy
2. zobrazujeme RGB, tedy bez max. přes kanály
3. není skutečný gradient, ale tzv. "guided"

výsledky pro vrstvu 6



# Guided backprop



velmi podobné jako saliency mapy, ovšem s několika rozdíly:

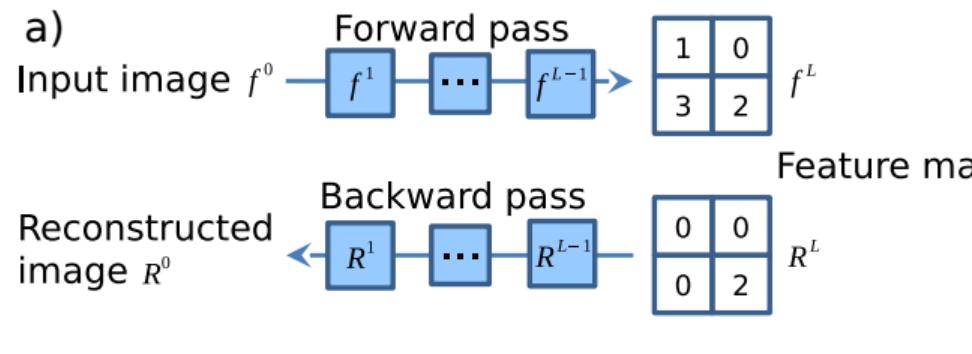
1. začneme backprop ne na vrchu sítě, ale na zvolené vrstvě
2. zobrazujeme RGB, tedy bez max. přes kanály
3. není skutečný gradient, ale tzv. "guided"

výsledky pro vrstvu **9**



# Guided backprop

“guided” = nepočítáme skutečný gradient; u RELU filtruje záporné gradienty  
proč? prostě to pak vypadá lépe ☺

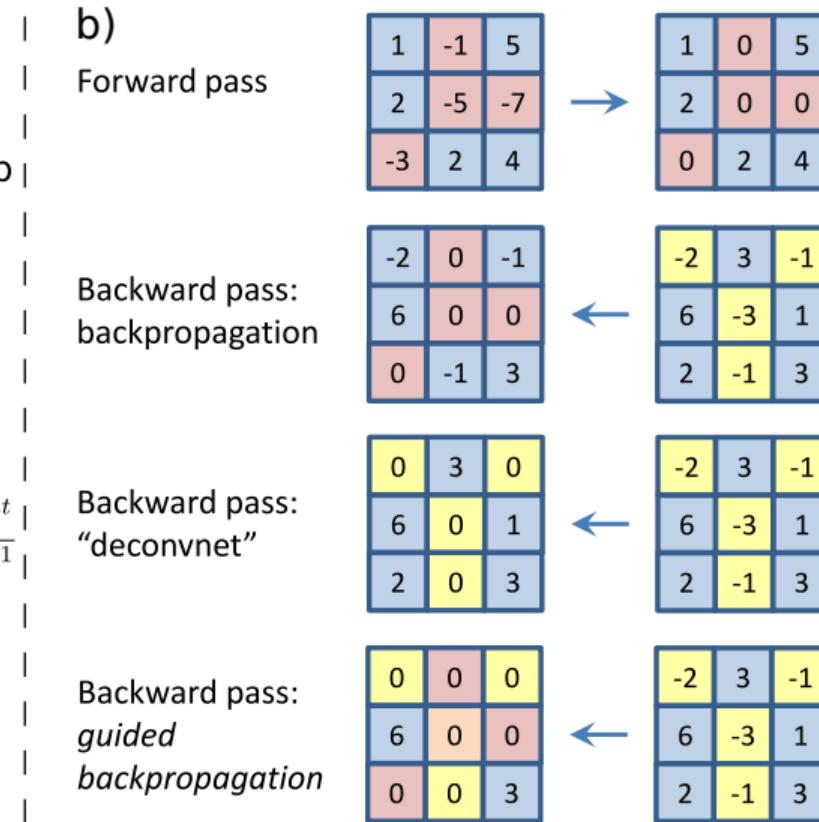


c) activation:  $f_i^{l+1} = \text{relu}(f_i^l) = \max(f_i^l, 0)$

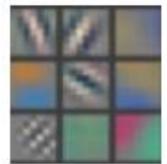
backpropagation:  $R_i^l = (f_i^l > 0) \cdot R_i^{l+1}$ , where  $R_i^{l+1} = \frac{\partial f^{out}}{\partial f_i^{l+1}}$

backward 'deconvnet':  $R_i^l = (R_i^{l+1} > 0) \cdot R_i^{l+1}$

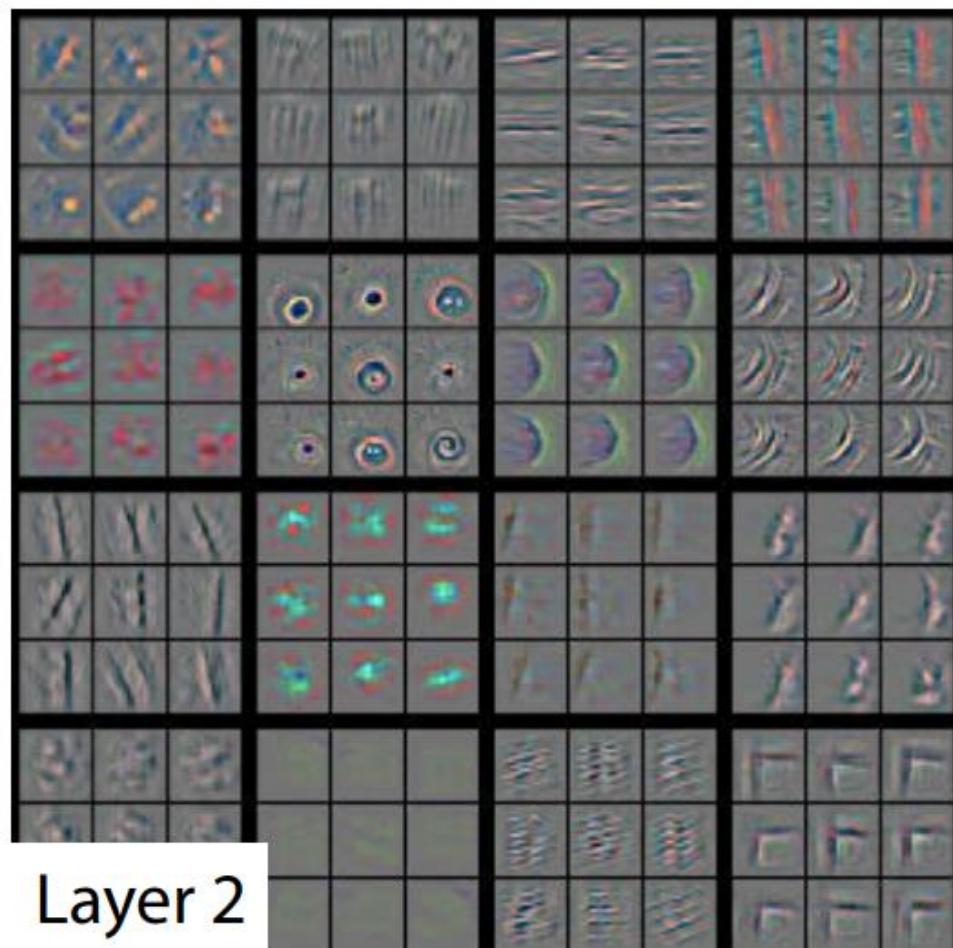
guided backpropagation:  $R_i^l = (f_i^l > 0) \cdot (R_i^{l+1} > 0) \cdot R_i^{l+1}$



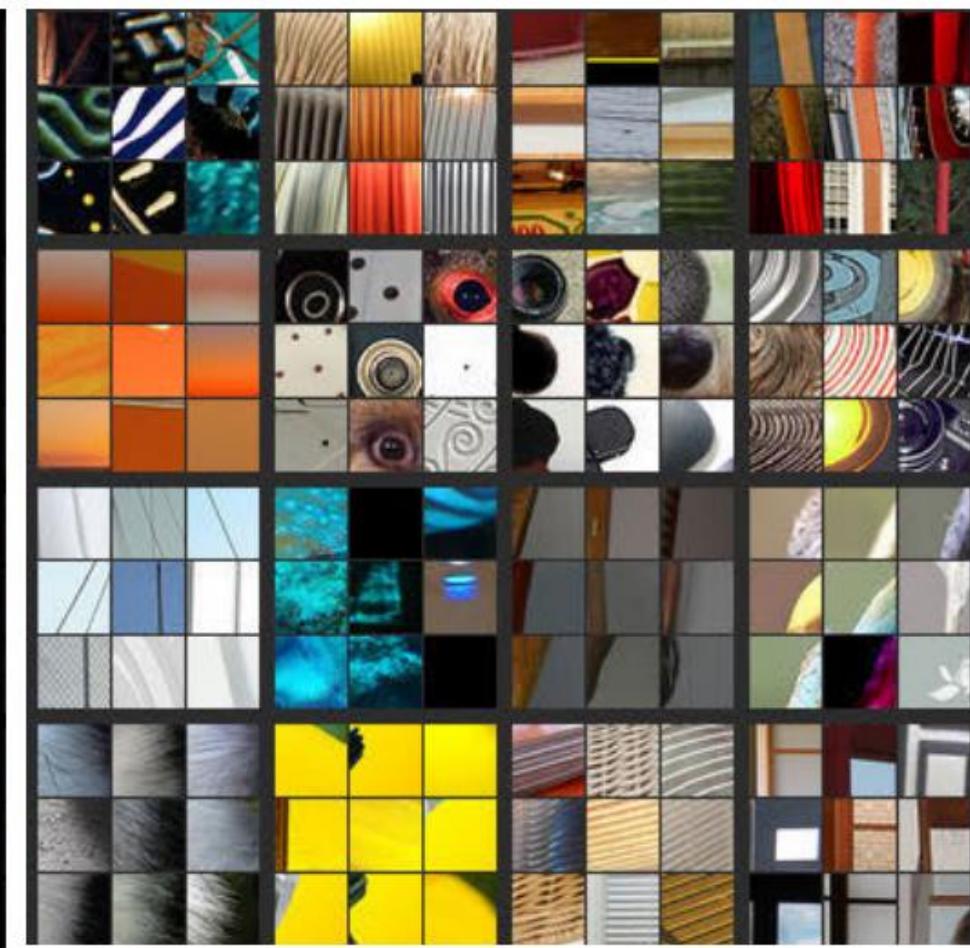
# Další příklady



Layer 1

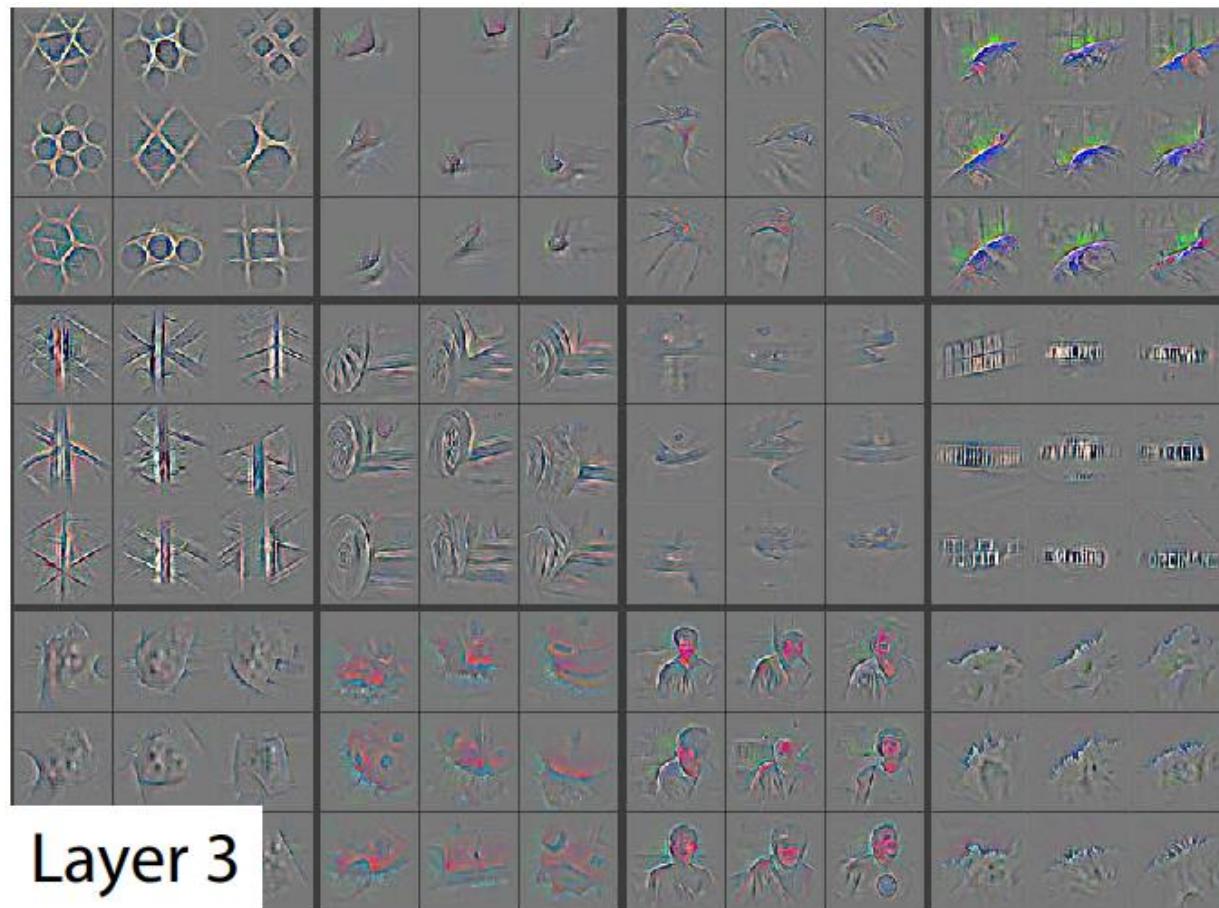


Layer 2



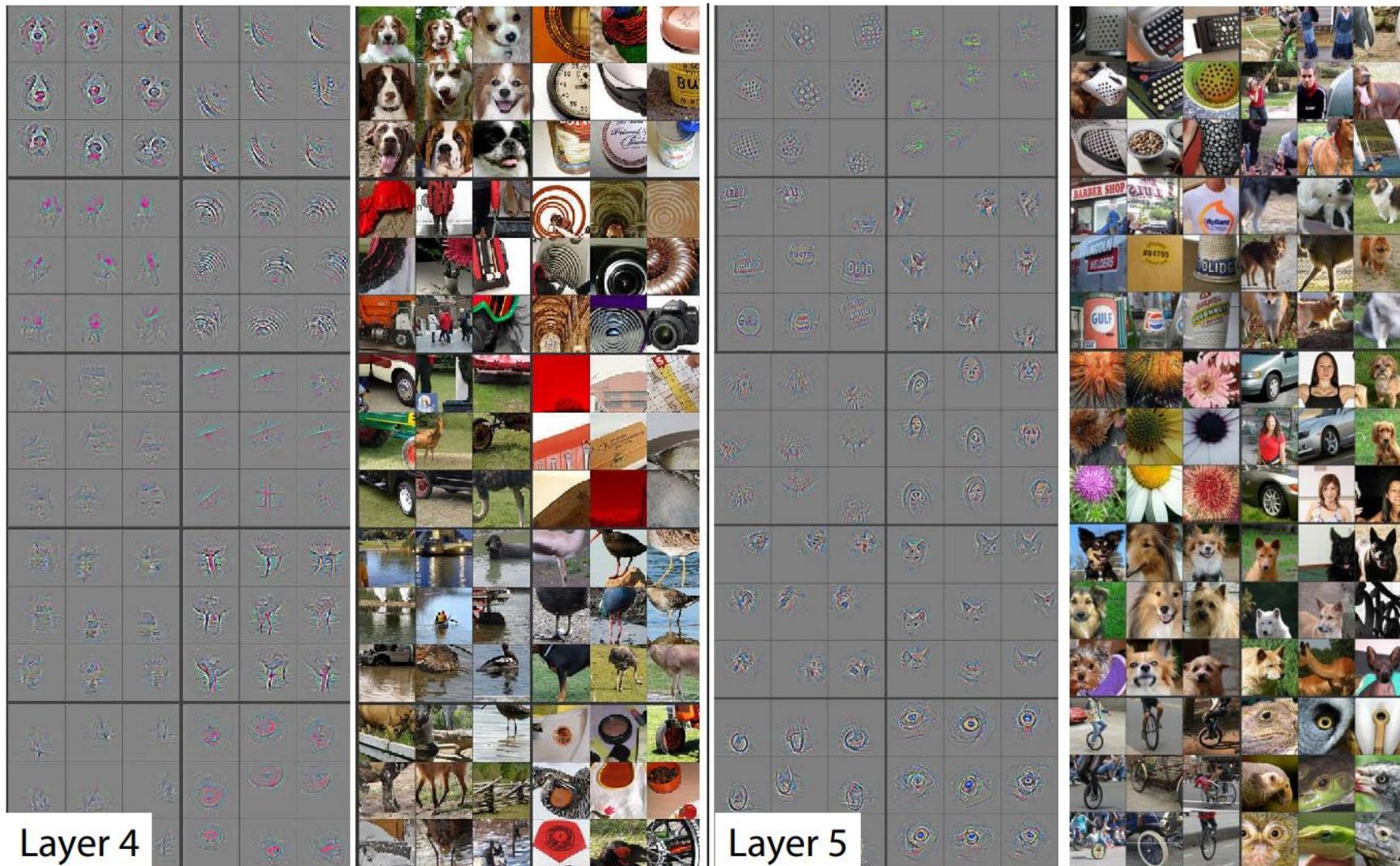
článek: [Zeiler, Fergus: Visualizing and Understanding Convolutional Networks](#)

## Další příklady



článek: [Zeiler, Fergus: Visualizing and Understanding Convolutional Networks](#)

# Další příklady



článek: [Zeiler, Fergus: Visualizing and Understanding Convolutional Networks](#)

# Vizualizace optimalizací (gradient ascent)

úloha: zkusme najít obrázek, který maximalizuje skóre s nějaké třídy  $c$

$$I^* = \arg \max_I s_c(I) + R(I)$$

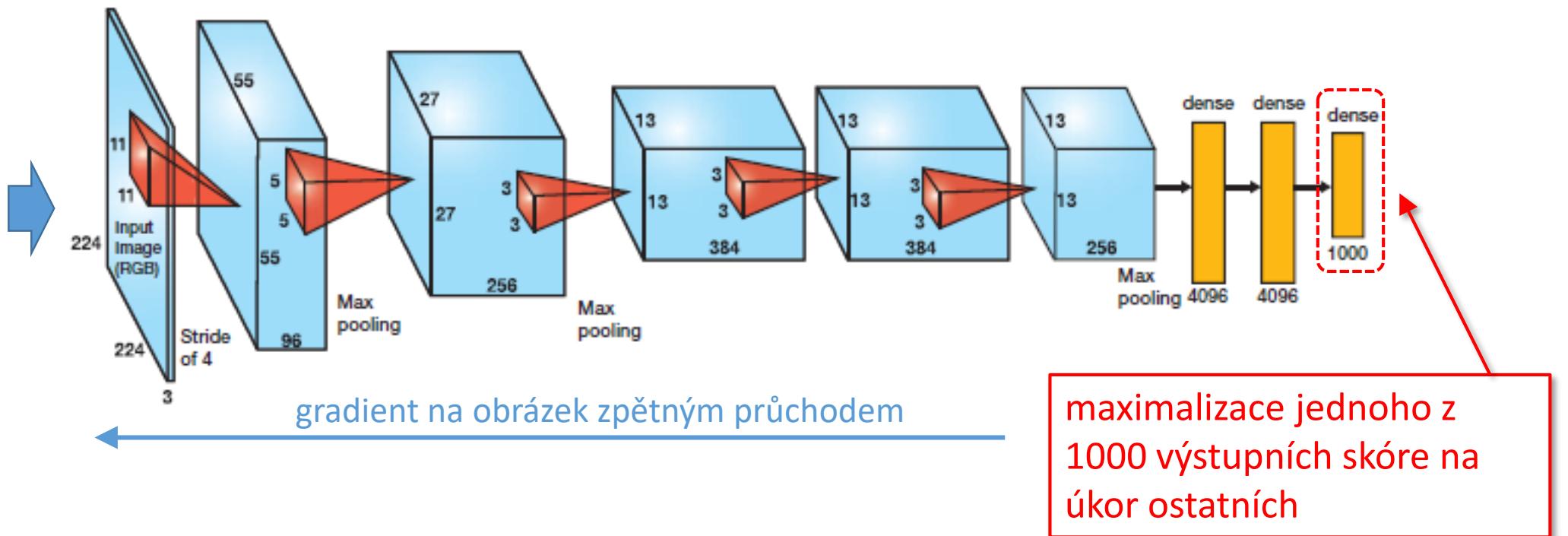
(guided) backprop: zobrazujeme derivaci nějakého neuronu vzhledem ke vstupnímu obrázku

optimalizace: postupně gradientem updatujeme obrázek + regularizujeme

regularizace např. L2:  $R(I) = \|I\|^2$

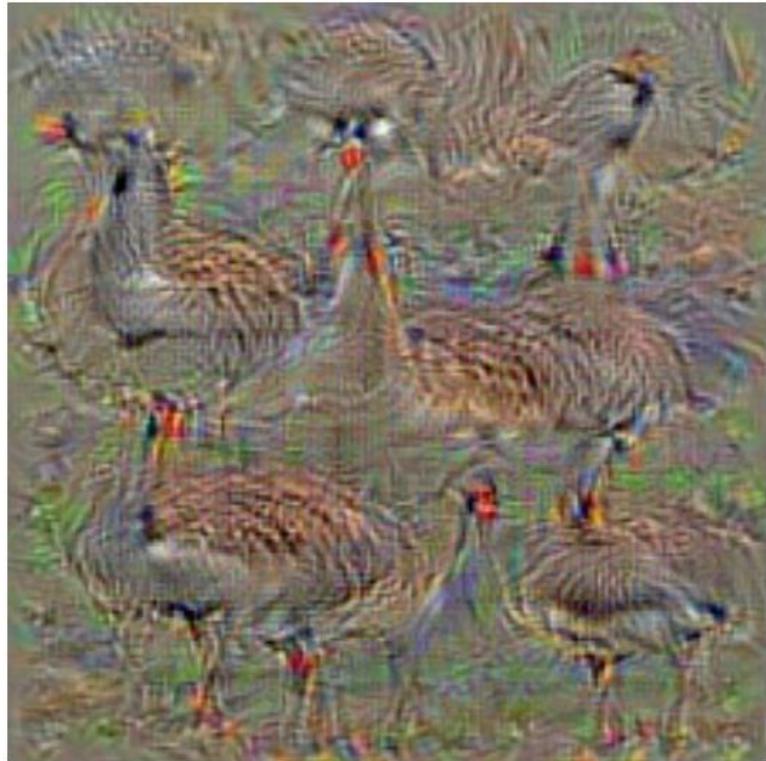
# Postup optimalizace

optimalizovaný  
obrázek



1. inicializace na nuly nebo náhodně
2. místo lossu nastavíme gradient shora na samé nuly kromě naší třídy  $[0, 0, \dots, 1, \dots, 0]$
3. zpětný průchod – opět nás zajímá gradient na vstup, ne na váhy
4. updatujeme vstup graidentem
5. zpět na 2. a opakujeme

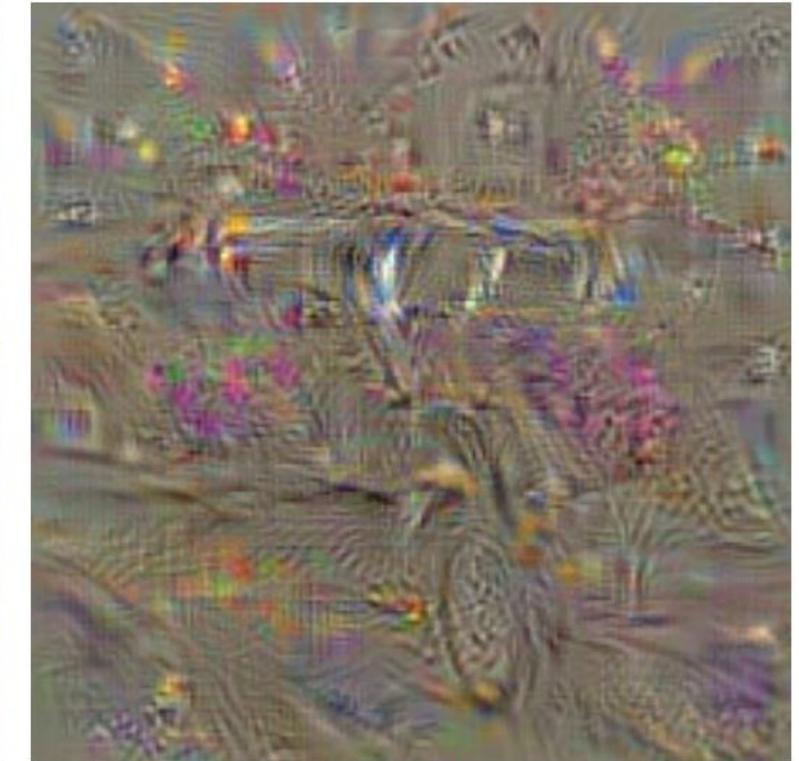
# Příklady obrázků maximalizujících zadanou třídu



**goose**



**ostrich**



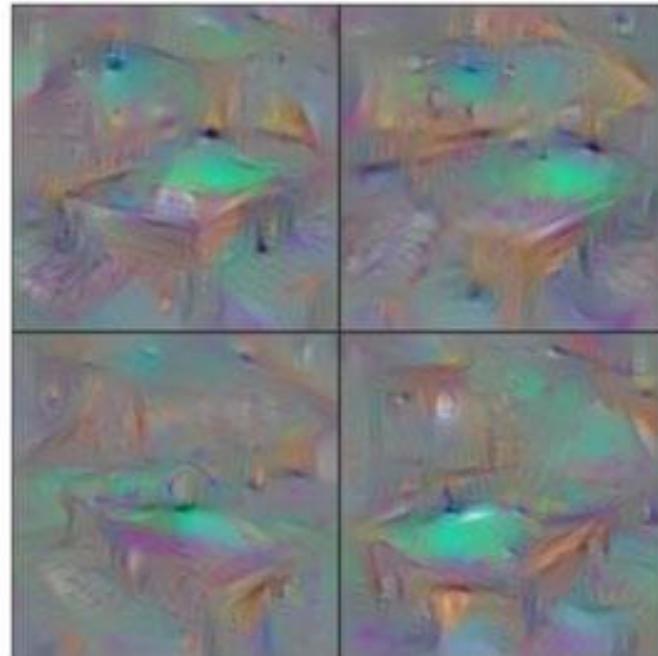
**limousine**

# Příklady obrázků maximalizujících zadanou třídu

lepší způsob regularizace, navíc vyhlazování (postprocessing) v každém kroku



Flamingo

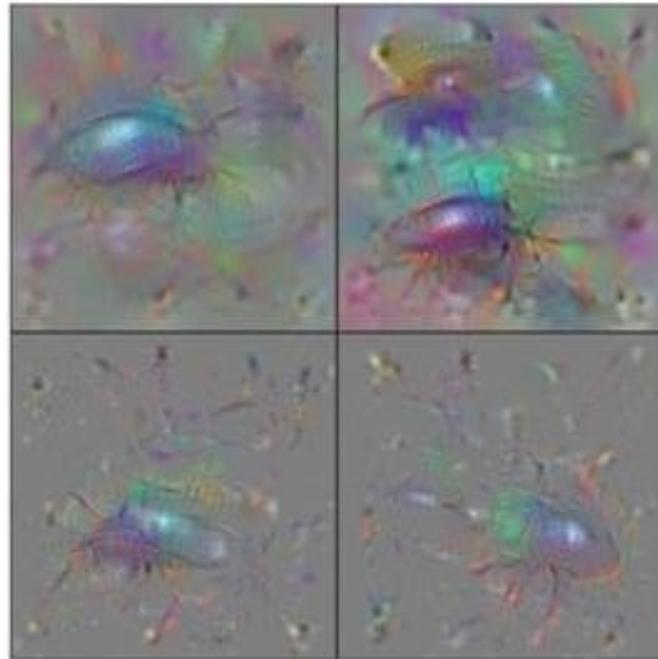


Billiard Table

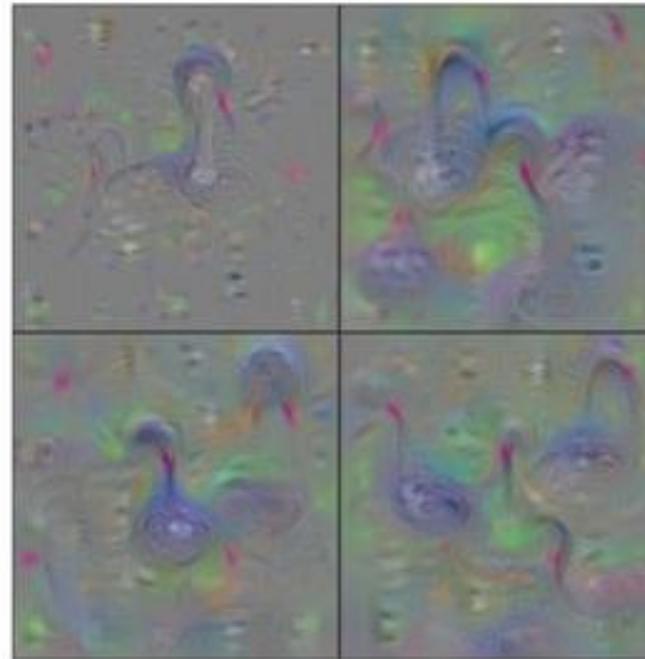


School Bus

# Příklady obrázků maximalizujících zadanou třídu



Ground Beetle



Black Swan

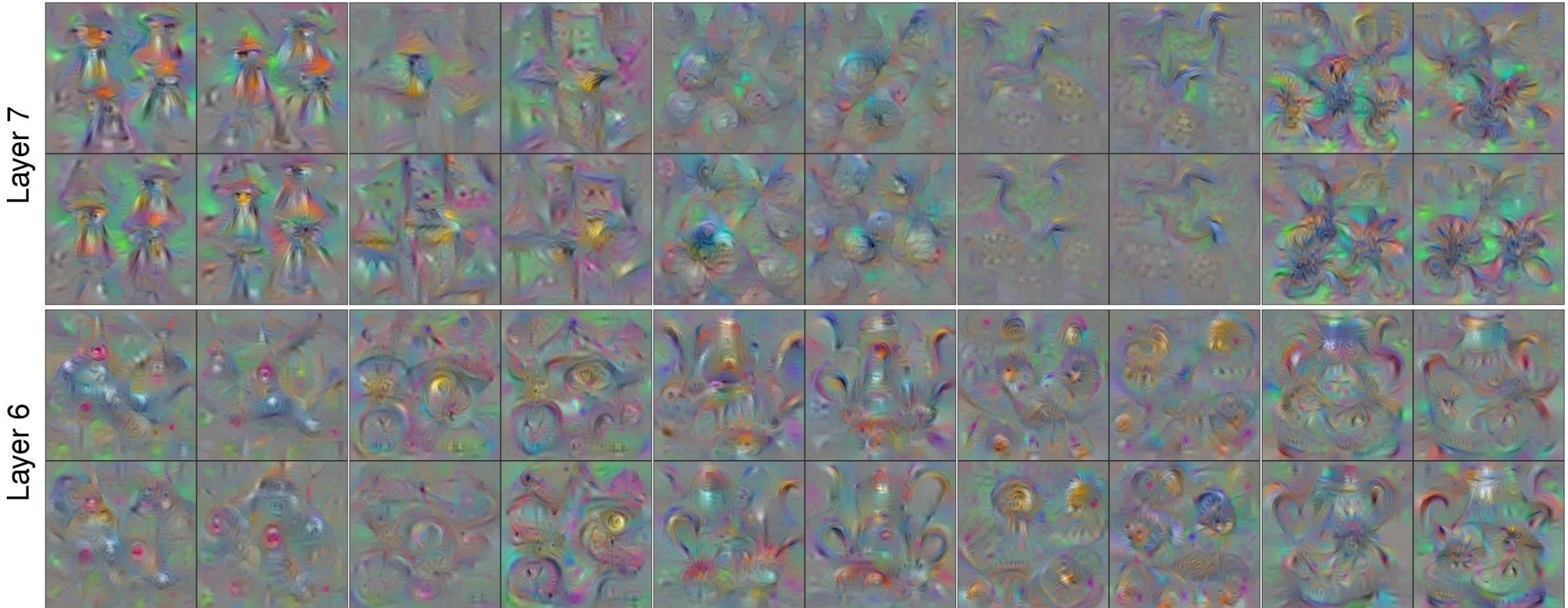


Tricycle

# Vizualizace vnitřních vrstev pomocí optimalizace

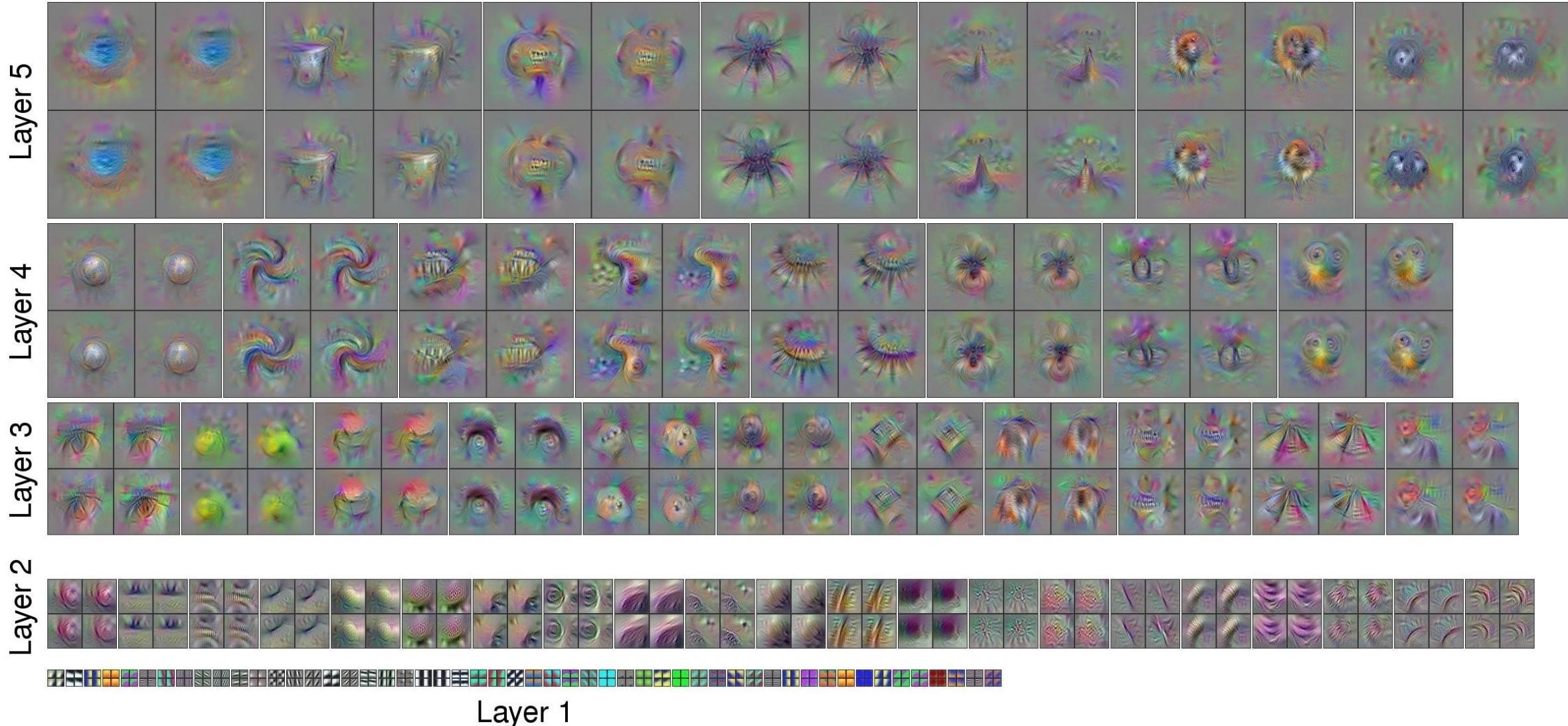


# Vizualizace vnitřních vrstev pomocí optimalizace



článek: [Yosinski et al.: Understanding Neural Networks Through Deep Visualization](#)

# Vizualizace vnitřních vrstev pomocí optimalizace



článek: [Yosinski et al.: Understanding Neural Networks Through Deep Visualization](#)

# Deep Dream

- Postup opět optimalizace
- Gradient nastaven na aktivace
- Pokud je aktivace velká, bude velký i gradient
- Malá aktivace → malý gradient
- Nesoustředí se na jeden neuron, ale na více zároveň
- Sítě na obrázku zesílí silné
- Pro dosažení hezkých výsledků se jako regularizace s každým průchodem náhodně mírně posune (jitter shift)



<https://github.com/google/deepdream>

# Deep Dream



Anemone Fish



Banana



Parachute



Screw



"Admiral Dog!"



"The Pig-Snail"



"The Camel-Bird"

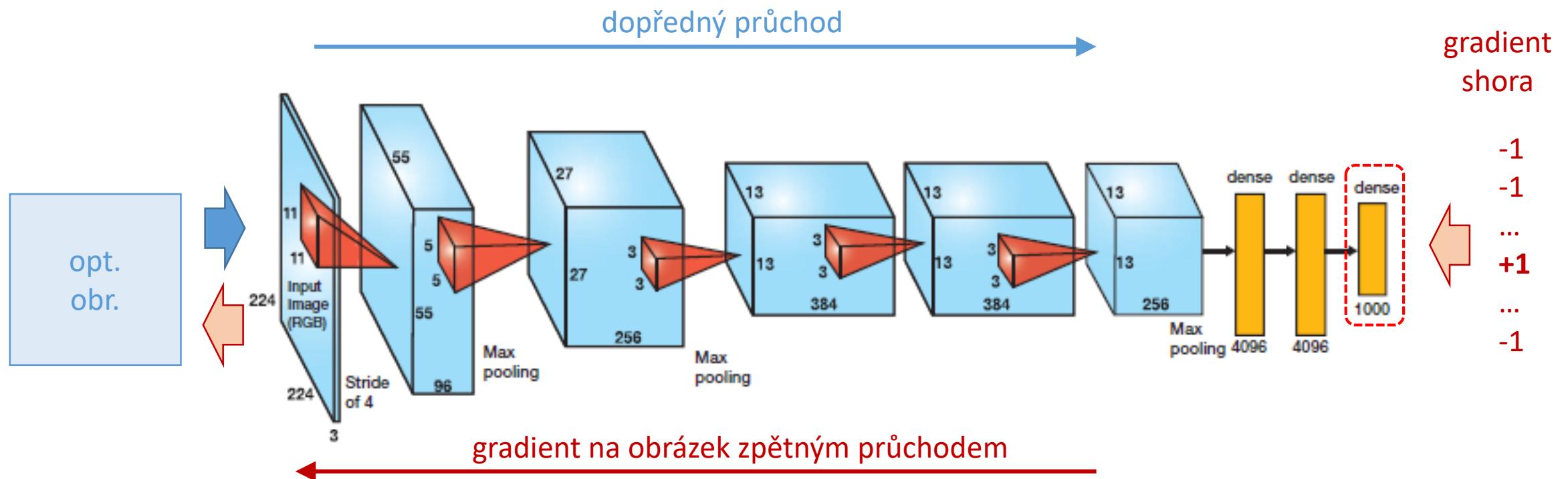


"The Dog-Fish"

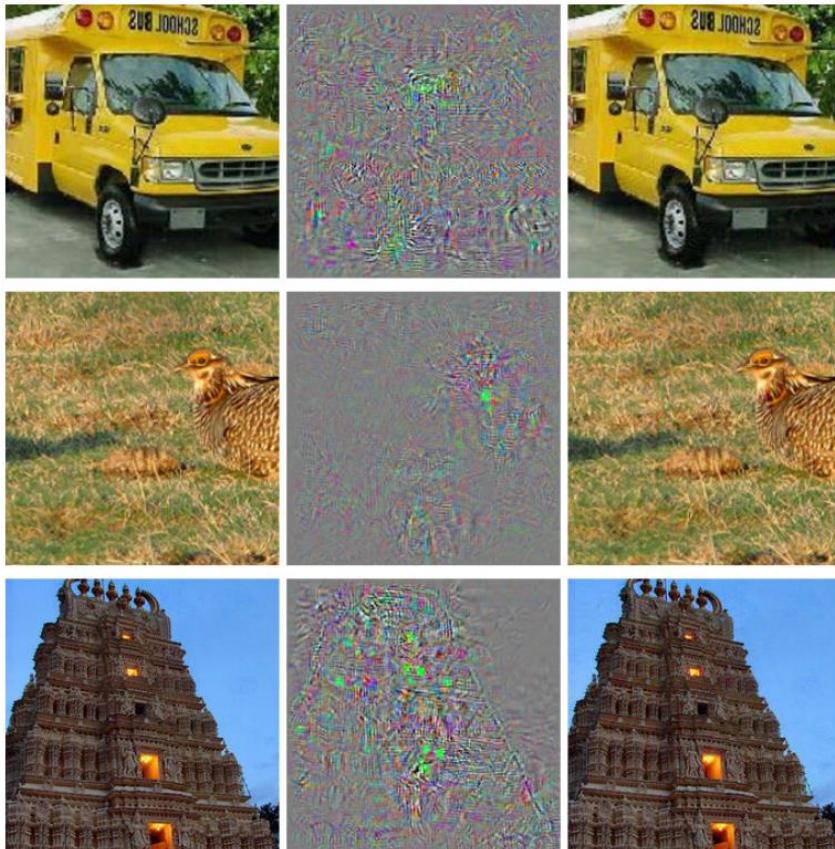
# Adversarial examples

- Optimalizaci na obrázek lze využít nejen pro vizualizaci, ale také k ovlivnění rozhodování sítě
- Co kdybychom např.:
  1. vzali nějaký obrázek a nechali projít sítí
  2. nastavili gradient tak, aby síť maximalizovala skóre jiné třídy než té správné
  3. vypočítali gradient na vstupní obrázek
  4. updatovali
  5. opakovali postup, dokud síť obrázek nezačne klasifikovat, jak si přejeme?

# Adversarial examples



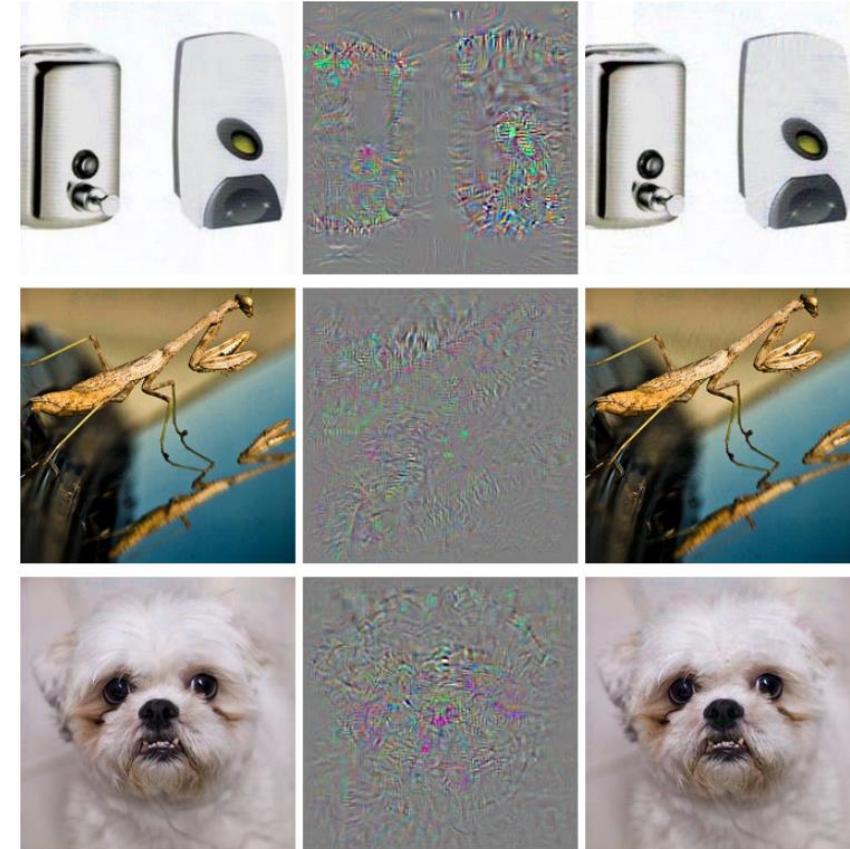
# Adversarial examples



správně

+ rozdíl

= pštros



správně

+ rozdíl

= pštros

# Adversarial examples

pudl ambulance basketbalový elektrická  
míč kytara

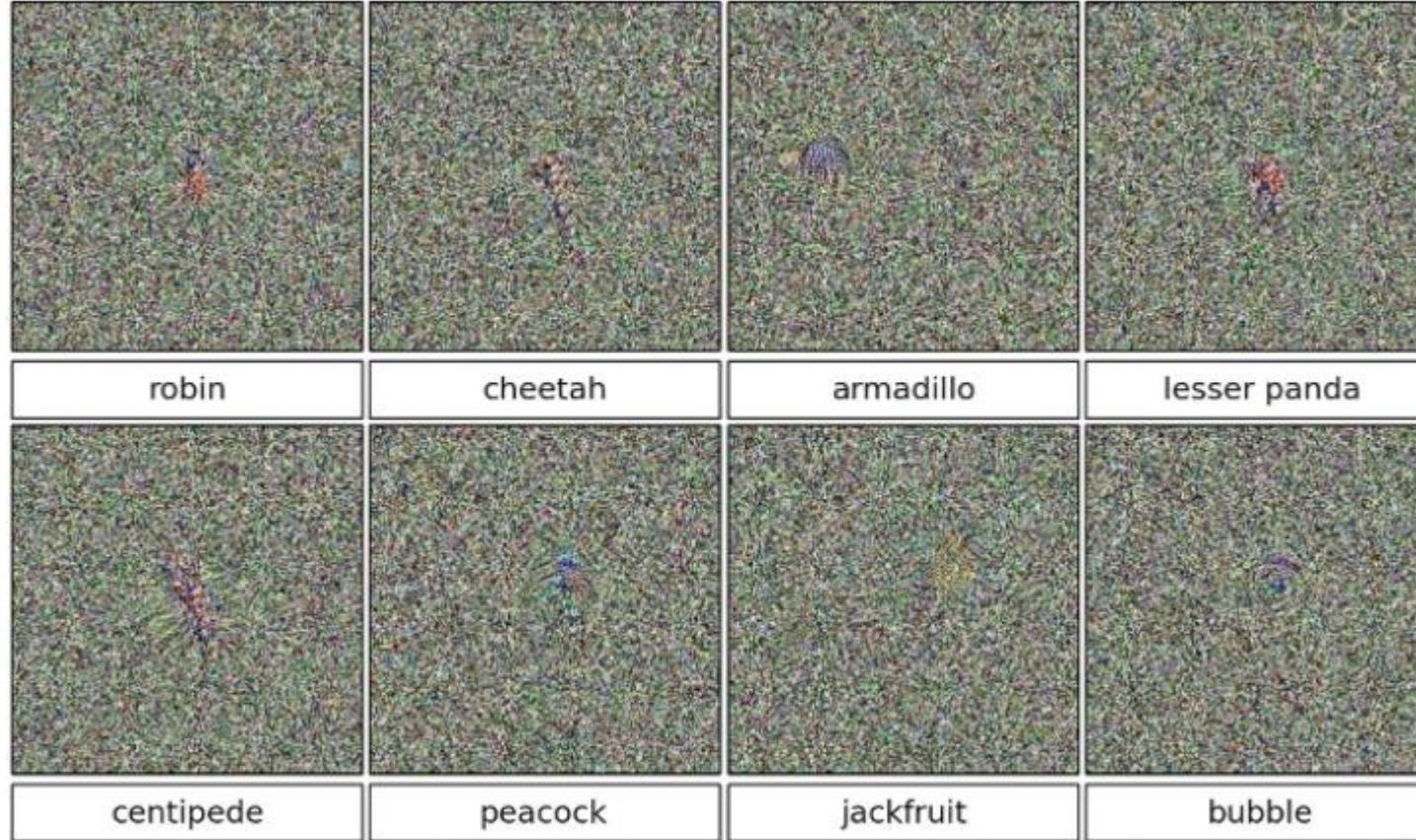


### (a) Adversarial examples

(b) Corresponding perturbations (amplified by 10 $\times$ )

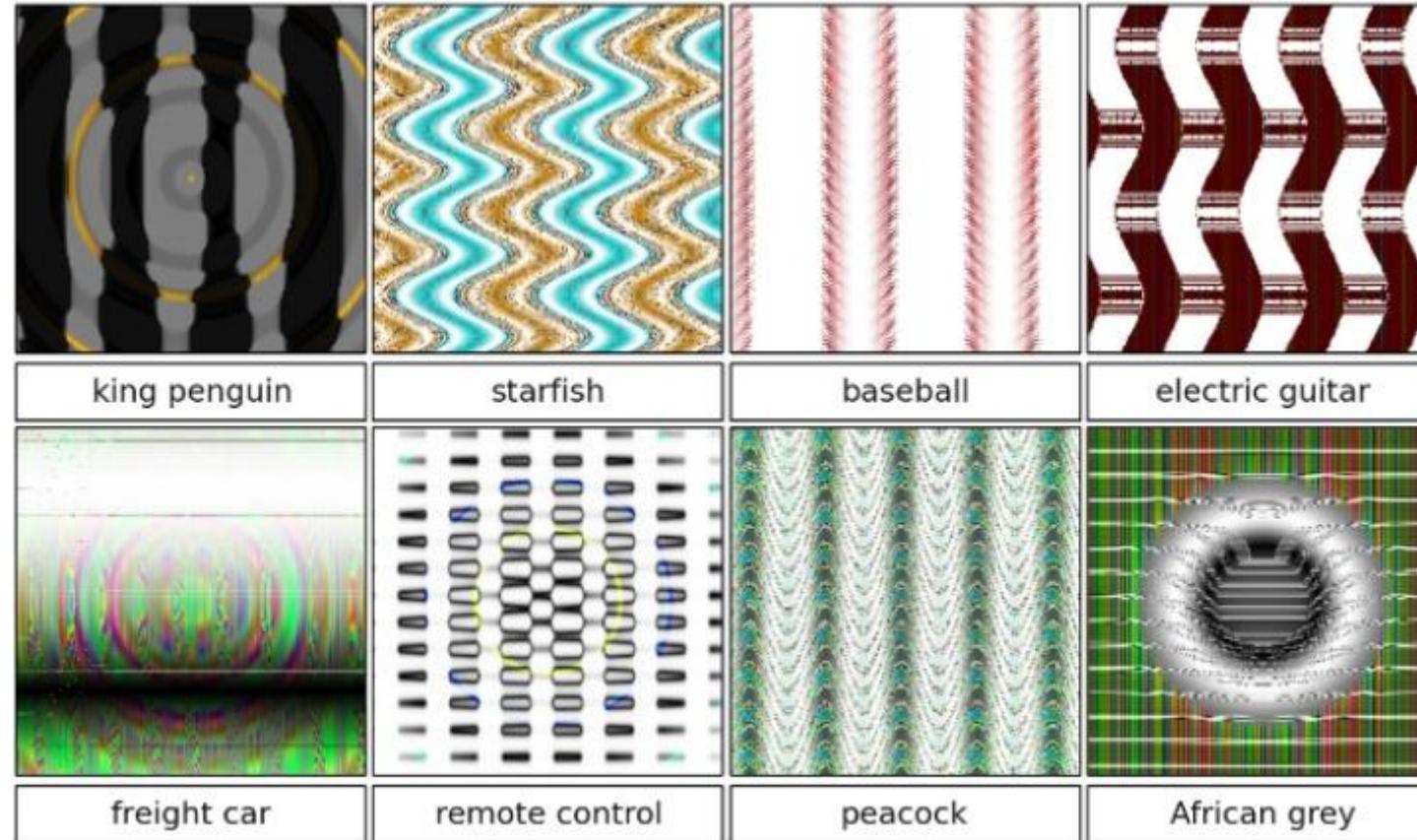
zdroj: Xiao et al: Generating adversarial examples with adversarial networks

# Adversarial examples



zdroj: [Nguyen et al: Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images](#)

# Adversarial examples



zdroj: [Nguyen et al: Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images](#)

# Proč to funguje

X	2	-1	3	-2	2	2	1	-4	5	1
w	-1	-1	1	-1	1	-1	1	1	-1	1

← input example  
← weights

class 1 score = dot product:

$$= -2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3$$

$$\Rightarrow \text{probability of class 1 is } 1/(1+e^{-(-3)}) = 0.0474$$

i.e. the classifier is **95%** certain that this is class 0 example.

$$P(y = 1 | x; w, b) = \frac{1}{1 + e^{-(w^T x + b)}} = \sigma(w^T x + b)$$

# Proč to funguje

X	2	-1	3	-2	2	2	1	-4	5	1
W	-1	-1	1	-1	1	-1	1	1	-1	1
adversarial x	?	?	?	?	?	?	?	?	?	?

← input example  
← weights

class 1 score = dot product:

$$= -2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3$$

=> probability of class 1 is  $1/(1+e^{-(-3)}) = 0.0474$

i.e. the classifier is **95%** certain that this is class 0 example.

$$P(y=1 | x; w, b) = \frac{1}{1 + e^{-(w^T x + b)}} = \sigma(w^T x + b)$$

# Proč to funguje

abychom zvýšili skóre:

ke každému prvku  $x_i$ , jemuž odpovídá *kladná* váha  $w_i$  přičteme 0.5

od každého prvku  $x_i$ , jemuž odpovídá *záporná* váha  $w_i$  odečteme 0.5

X	2	-1	3	-2	2	2	1	-4	5	1
W	-1	-1	1	-1	1	-1	1	1	-1	1
adversarial x	1.5	-1.5	3.5	-2.5	2.5	1.5	1.5	-3.5	4.5	1.5

← input example  
← weights

class 1 score before:

$$-2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3$$

=> probability of class 1 is  $1/(1+e^{(-(-3))}) = 0.0474$

$$\textcolor{red}{-1.5+1.5+3.5+2.5+2.5-1.5+1.5-3.5-4.5+1.5 = 2}$$

=> probability of class 1 is now  $1/(1+e^{(-(2)}) = 0.88$

i.e. we improved the class 1 probability from 5% to 88%

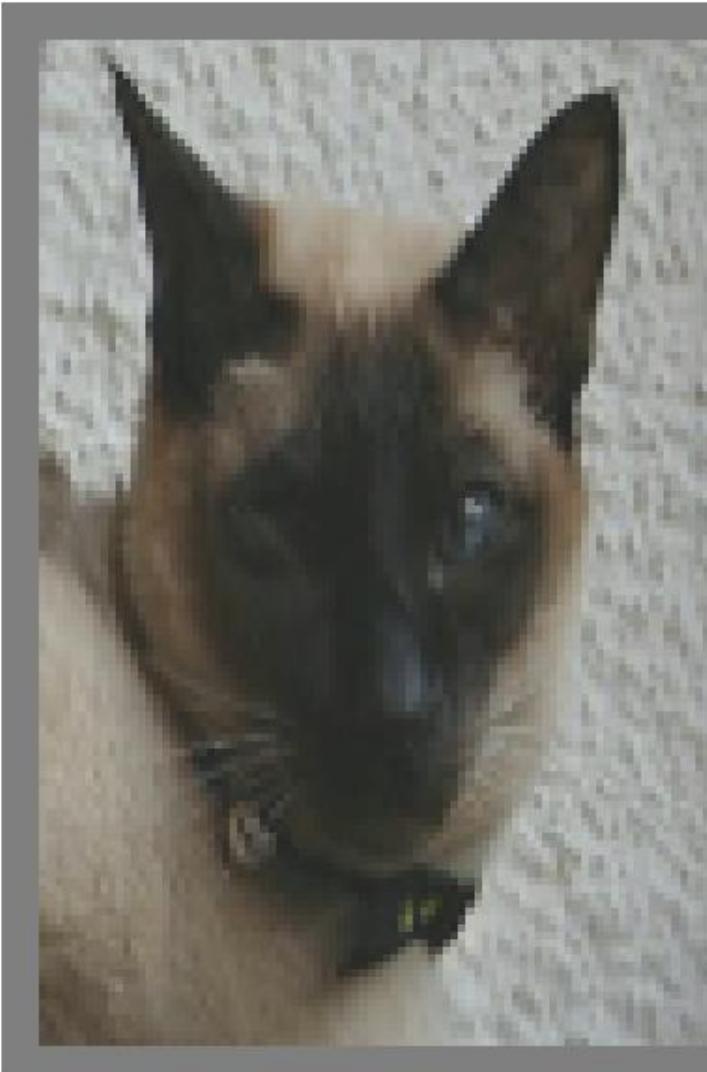
$$P(y=1 | x; w, b) = \frac{1}{1 + e^{-(w^T x + b)}} = \sigma(w^T x + b)$$

# Adversarial examples

- Jsou bezpečnostní hrozbou? Ano i ne.
- Změna obrázku musí být cílená – náhodná perturbace se takto chovat nebude
- Nezáleží, zda je klasifikátor lineární či nelineární
- Dokonce funguje i pro jiné typy než lineární klasifikátory, např. nejbližší soused atd.
- Problém je především prostor mimo oblast, na které se síť učila
- **Různé sítě natrénované na stejné úloze (např. ImageNet), mají stejné adversarialy!**
  - příp. útočník tak nepotřebuje přítup k vaší síti
  - [Szegedy et al: Intriguing properties of neural networks](#)
- Obrana vůči adversarialům je aktivní oblast výzkumu

# Adversarial examples, které zmatou i člověka

kočka



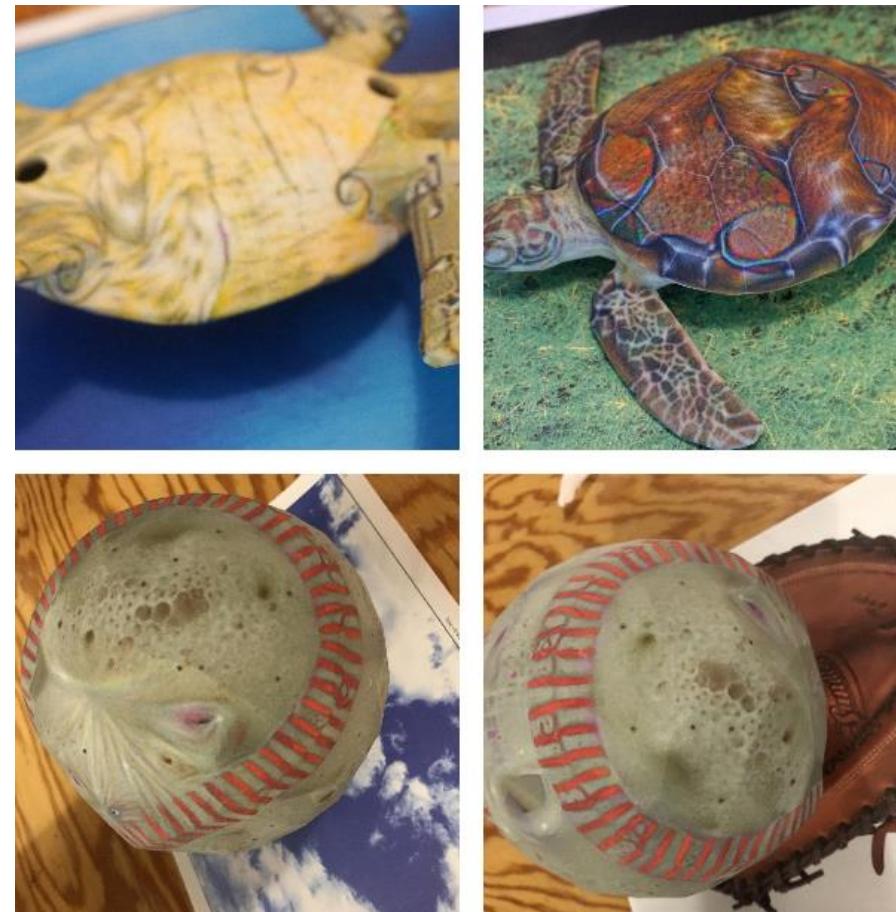
pes



článek: [Elsayed et al: Adversarial Examples that Fool both Human and Computer Vision](#)

# Fyzické objekty jako adversarial examples

objekty klasifikované jako puška (rifle)



zdroj: [Athalye et al: Synthesizing Robust Adversarial Examples](#)

# Neural style transfer

Úloha: přetrasformovat styl obrázku do jiného tak, aby např. vypadal jako malba



+



=



obrázek

styl

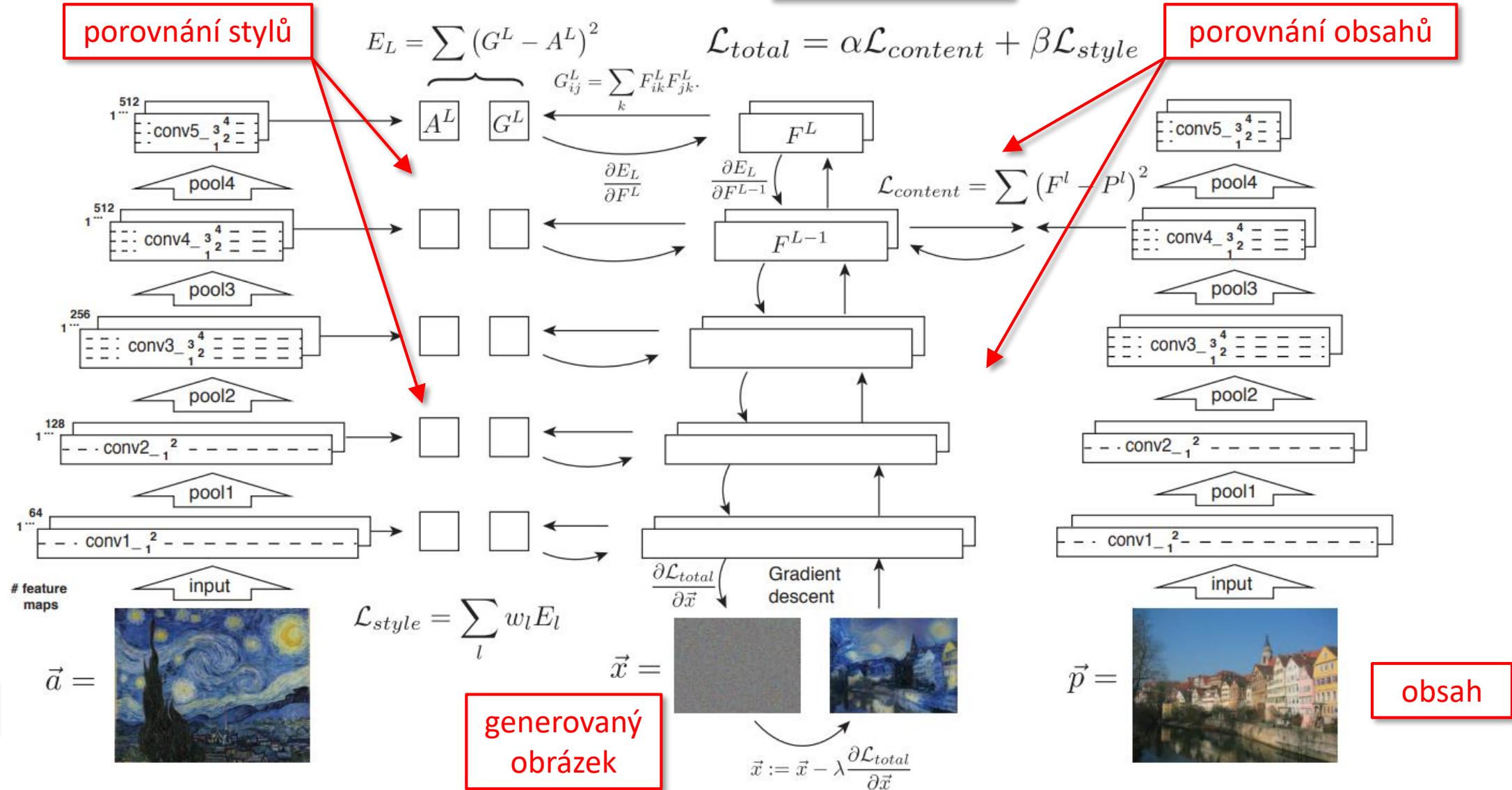
výsledek

článek: [Gatys et al: Image Style Transfer Using Convolutional Neural Networks](#)

# Transfer stylu

- Opět jako optimalizace obrázku
- Chceme upravit vstup tak, aby měl
  1. stejný obsah (content)
  2. stejný styl (style)
- Obsah ohodnotíme skrze konvoluční mapy, tj. budeme porovnávat konvoluční mapy vstpu a reference → content loss
- Ale co styl?

# Transfer stylu



článek: [Gatys et al: Image Style Transfer Using Convolutional Neural Networks](#)

# Gram matrix

gram matrix = vzájemná korelace filtrů

$F^L$  je konvoluční mapa vrstvy  $L$

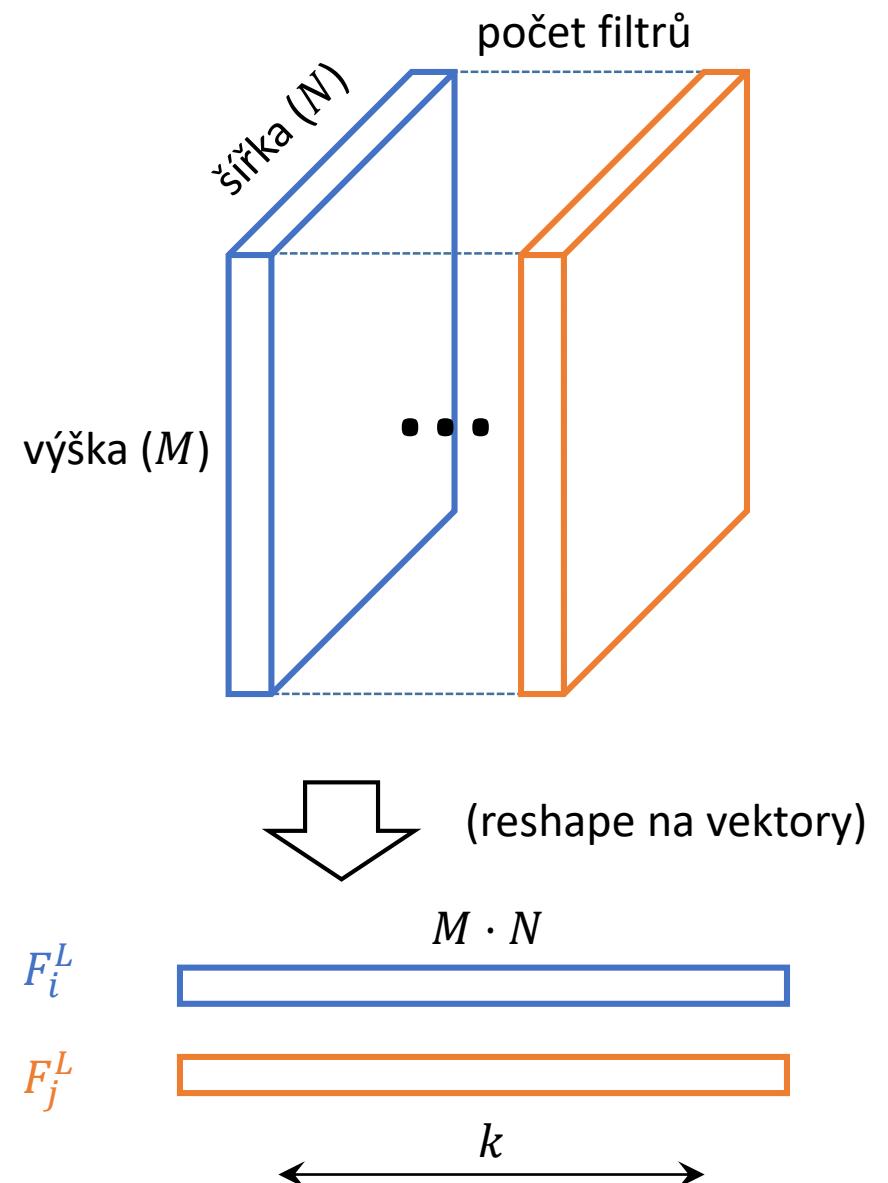
$$G_{ij}^L = \sum_k F_{ik}^L F_{jk}^L$$

*i, j jsou indexy dvou různých filtrů*

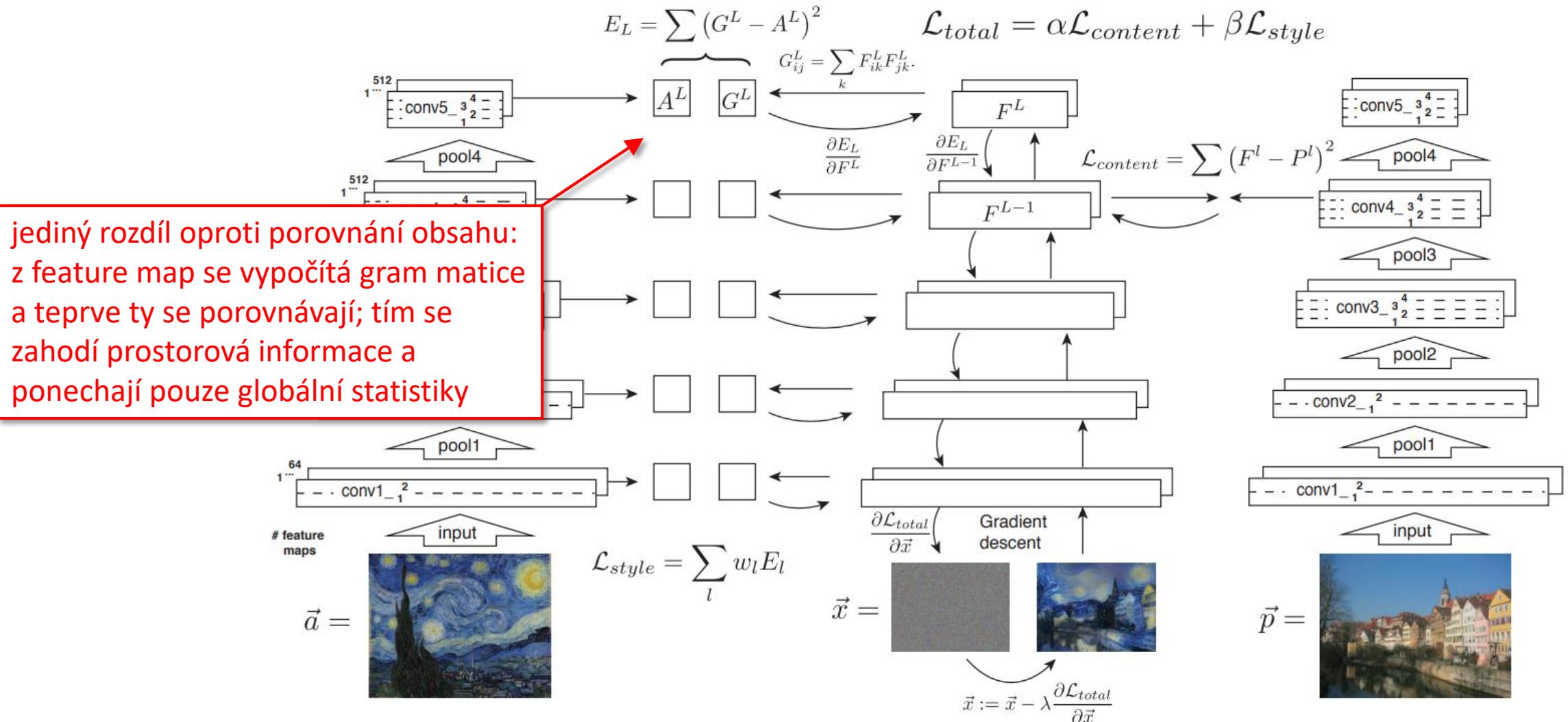
$G_{ij}$  je matice o rozměrech  
počet filtrů x počet filtrů  
(párové korelace filtrů vrstvy L)

$k$  jde přes všechny pixely: uvažuje se, že kanály mapy  $F^L$  jsou z  $M \times N$  převedeny do vektoru o délce  $M \cdot N$

numpy:  
`F = F.reshape(F.shape[0], -1)`  
`G = np.dot(F, F.T)`



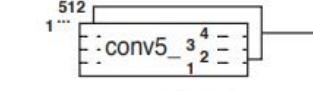
# Porovnání stylu



článek: [Gatys et al: Image Style Transfer Using Convolutional Neural Networks](#)

# Porovnání stylu

$$E_L = \sum (G^L - A^L)^2 \quad \mathcal{L}_{total} = \alpha \mathcal{L}_{content} + \beta \mathcal{L}_{style}$$



# Ukázky neural style transferu

A

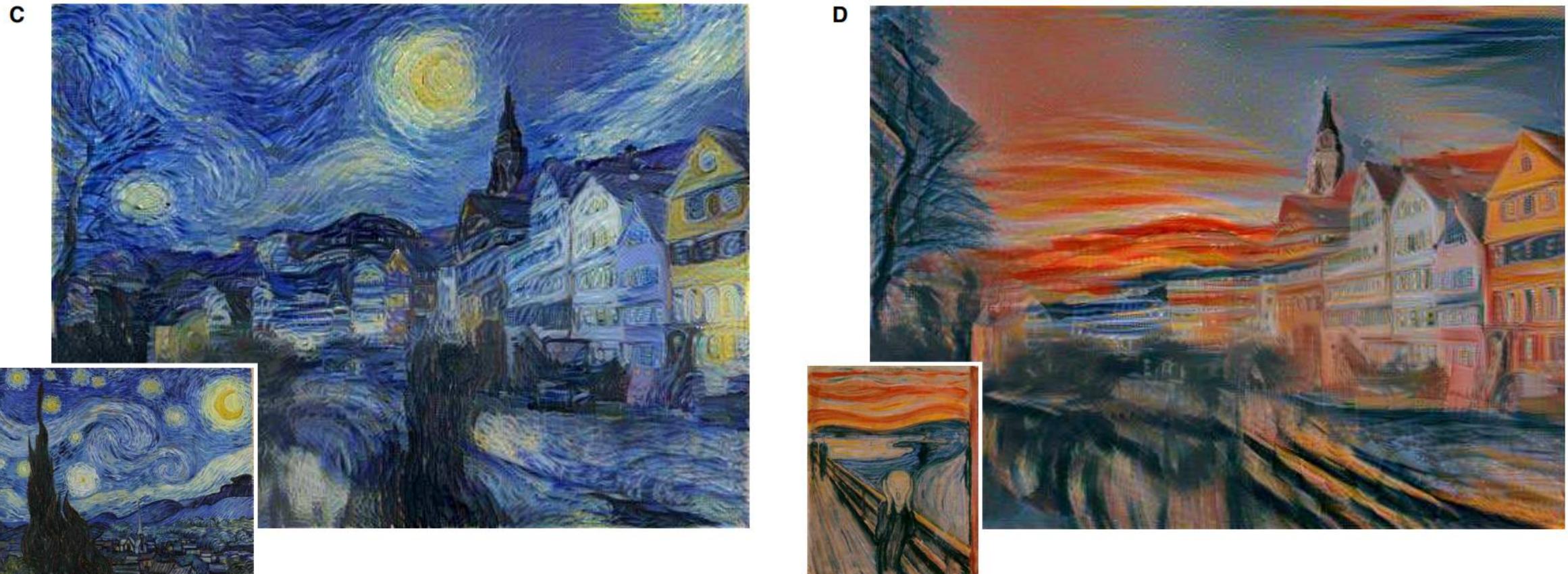


B



článek: [Gatys et al: Image Style Transfer Using Convolutional Neural Networks](#)

# Ukázky neural style transferu



článek: [Gatys et al: Image Style Transfer Using Convolutional Neural Networks](#)

# Ukázky neural style transferu

E



F



článek: [Gatys et al: Image Style Transfer Using Convolutional Neural Networks](#)