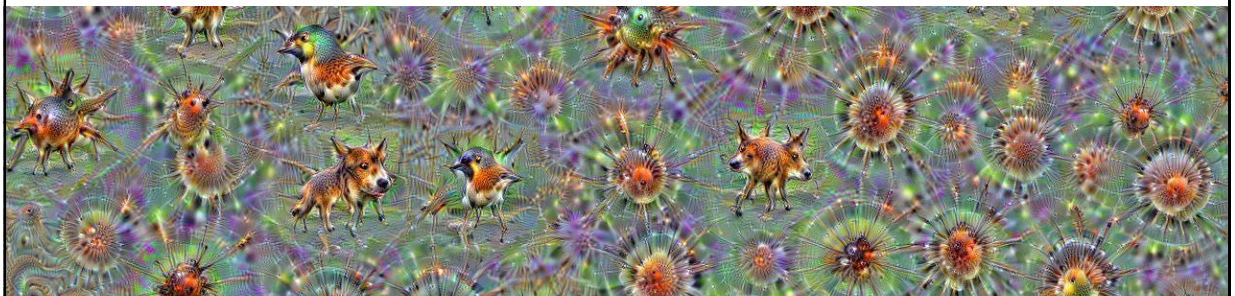
TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY



Module 12: Visualization and understanding

5LSM0: Convolutional neural networks for computer vision

Fons van der Sommen

Electrical Engineering / VCA research group

VCA Video Coding & Architectures research group

Administrative: Paper sessions schedule March 24st [OK?]

Time	Paper title	Presenters	Time	Paper title	Presenters
8:45 – 9:00	Mask R-CNN	David Matos Rodriguez Floris Straver	**10:30 – 10:45**	Image Style Transfer Using Convolutional Neural Networks	Adriano Cardace Frouke Hekker Jesse van Oort
9:00 – 9:15	Squeeze-and-Excitation Networks	Noud van de Geve Tim Schoonbeek Menno Nijssen	**10:45 – 11:00**	MultiResUNet : Rethinking the U-Net Architecture for Multimodal Biomedical Image Segmentation	Gino Jansen Max van Riel
9:15 – 9:30	Shake-shake regularization	Joël de Bruijn Jelle Westbeek Mark Legters	**11:00 – 11:15**	High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs	Shubham Ghatge Tim Houben
9:30 – 9:45	Dataset distillation	Thomas Bastiaansen Rick Butler	**11:15 – 11:30**	BREAK	
9:45 – 10:00	YOLO v3: an incremental improvement	Chuchen Cai Yawei Liu Xiang Zhang	**11:30 – 11:45**	Learning Transferable Architectures for Scalable Image Recognition (Neural Architecture Search)	Ties Hendrickx Koen Sanders
10:00 – 10:15	BREAK		**11:45 – 12:00**	Group Normalization	Genyu Song Lan Min
10:15 – 10:30	Progressive growing of GANs for improved quality, stability, and variation	Rishabh Kumar Gupta M P Smyl T O Kocsis	**12:00 – 12:15**	Social LSTM: Human Trajectory Prediction in Crowded Spaces.	Celine Vincent Gijs Cunnen
			12:15 – 12:30	Spatial Transformer Networks	Daan van den Hof

Administrative

Paper sessions

- What I at least expect in your presentation is:
  1) *description of the problem*
  2) *state-of-the-art alternatives + pros/cons*
  3) *proposed methodology + novelty of the approach*
  4) *results*
  5) *limitations*
  6) *conclusions*
- 10 minutes presentation, 3 minutes questions
- Please make sure each group member presents a part of the presentation
- Upload your slides before Wednesday March 23rd 23:59 CET through Canvas (assignment III)

Administrative

FINAL ASSIGNMENT

Automotive: Cityscapes segmentation challene

- https://www.cityscapes-dataset.com/benchmarks/#scene-labeling-task

Administrative

FINAL ASSIGNMENT

Automotive: Cityscapes segmentation challene

- https://www.cityscapes-dataset.com/benchmarks/#scene-labeling-task

Healthcare: ISIC 2018 Skin lesion classification

- https://challenge2018.isic-archive.com/task3/

Administrative

FINAL ASSIGNMENT

Automotive: Cityscapes segmentation challene

- https://www.cityscapes-dataset.com/benchmarks/#scene-labeling-task

Healthcare: ISIC 2018 Skin lesion classification

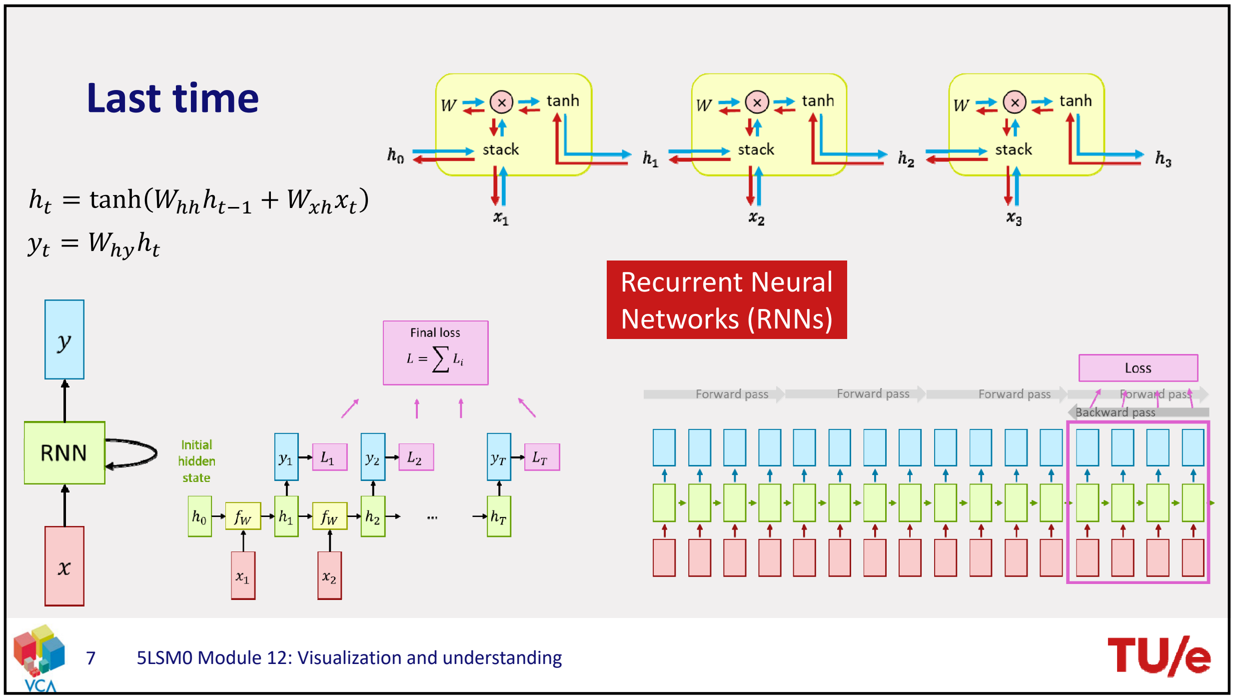
- https://challenge2018.isic-archive.com/task3/

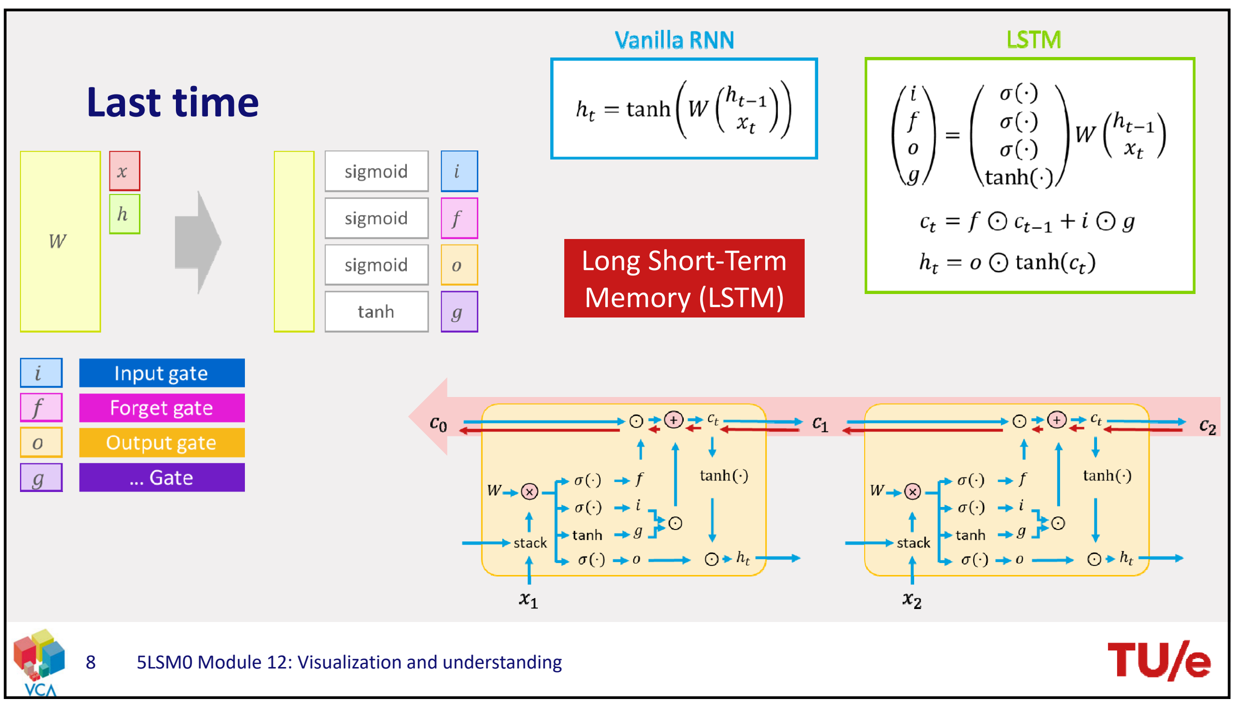
Bonus points for high scores / 10.0 for exceptional score

Write max 4-p paper double-column IEEE style

➢ ***More details will follow on Canvas!***

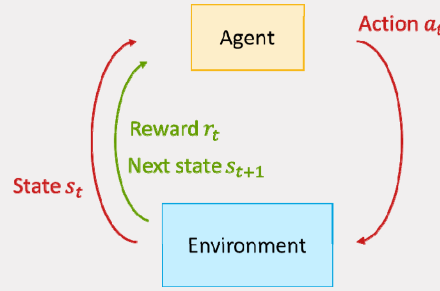
	140 hrs (5 ECTS)
Assignment I (10%)	14
Assignment II (10%)	14
Assignment III (10%)	14
Written exam (35%) *(incl lectures)*	49 *(24)*
Final assignment (35%)	49


Last time
$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$
$y_t = W_{hy}h_t$
W
tanh
stack
h_0
h_1
h_2
h_3
x_1
x_2
x_3
Recurrent Neural Networks (RNNs)
y
RNN
x
Final loss
$L = \sum L_i$
Initial hidden state
h_0
f_W
h_1
h_2
h_T
y_1
y_2
y_T
L_1
L_2
L_T
x_1
x_2
Loss
Forward pass
Backward pass
7
5LSM0 Module 12: Visualization and understanding
TU/e


Last time
Vanilla RNN
$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$
LSTM
$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma(\cdot) \\ \sigma(\cdot) \\ \sigma(\cdot) \\ \tanh(\cdot) \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$
$c_t = f \odot c_{t-1} + i \odot g$
$h_t = o \odot \tanh(c_t)$
W
x
h
sigmoid
sigmoid
sigmoid
tanh
i
f
o
g
Long Short-Term Memory (LSTM)
Input gate
Forget gate
Output gate
... Gate
c_0
c_1
c_2
x_1
x_2
stack
h_t
8
5LSM0 Module 12: Visualization and understanding
TU/e

Last time

Reinforcement learning



$\mathcal{S}$ Set of possible states
$\mathcal{A}$ Set of possible actions
$\mathcal{R}$ Distribution over reward, given state-action pair
$\mathbb{P}$ Transition probability: distribution over next state given state-action pair
γ Discount factor: how much do we value early vs. late rewards

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t \mid \pi\right] \quad \text{with} \quad s_0 \sim p(s_0), a_t \sim \pi(\cdot \mid s_t), s_{t+1} \sim \mathbb{P}(\cdot \mid s_t, a_t)$$

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s',a') \mid s,a\right]$$

This time

What is happening inside a convolutional neural network?

- What are the patterns to which certain layer respond?
- What patterns cause a particular neuron to fire maximally?
- What happens if we block out certain image parts?
- What is the effect of changing pixels in my image on the classification result?

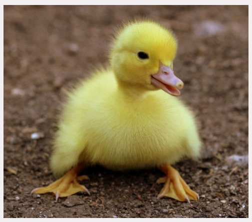
Can we exploit this knowledge for other purposes?

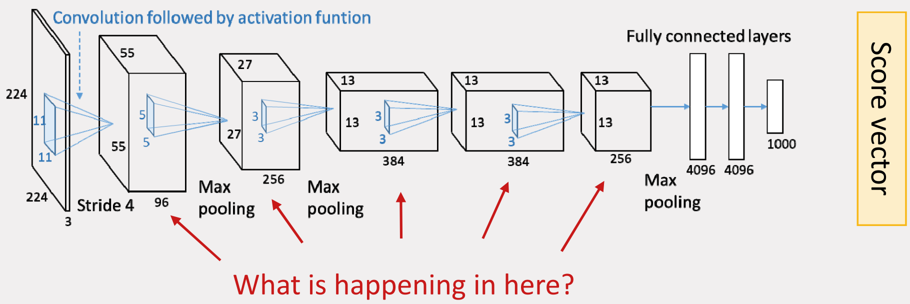
- Can we change the pixels in order to fool the network?
- Can we generate these patterns from the network?

Visualization

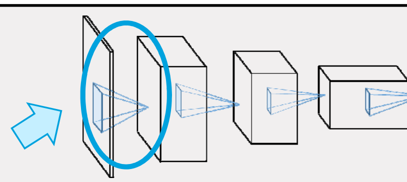
So far, neural network is more or less a black box

- We know how they are wired, we know how they're trained…
- But we're not really sure what happens inside that makes it work





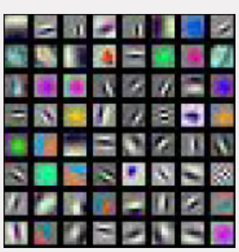
Visualization



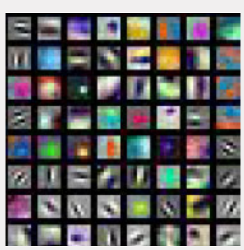
Q: Why can't we do the same thing for the other layers?
(as straightforwardly…)

Visualize filters of first conv layer

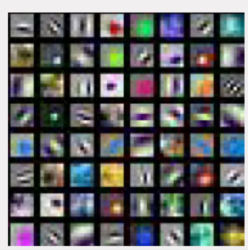
- Similar to responses in the mammalian brain!
- Similar to wavelets!



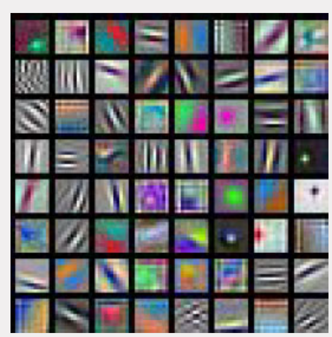
DenseNet-121
64×3×11×11



ResNet-18
64×3×11×11



ResNet-101
64×3×11×11



AlexNet:
64×3×11×11

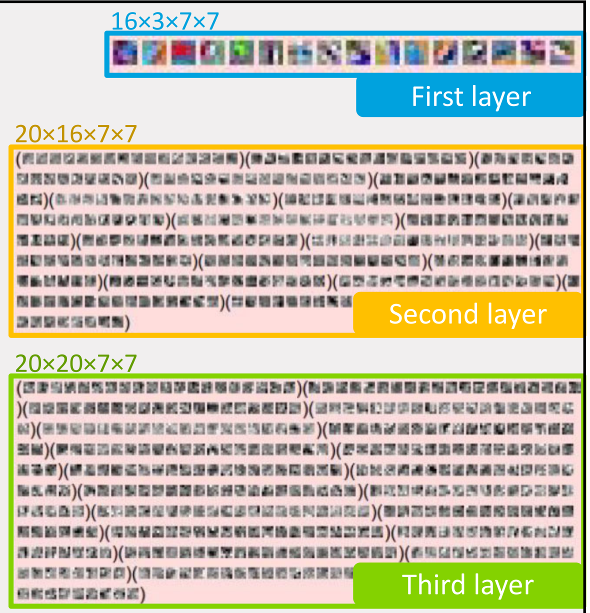
Visualization

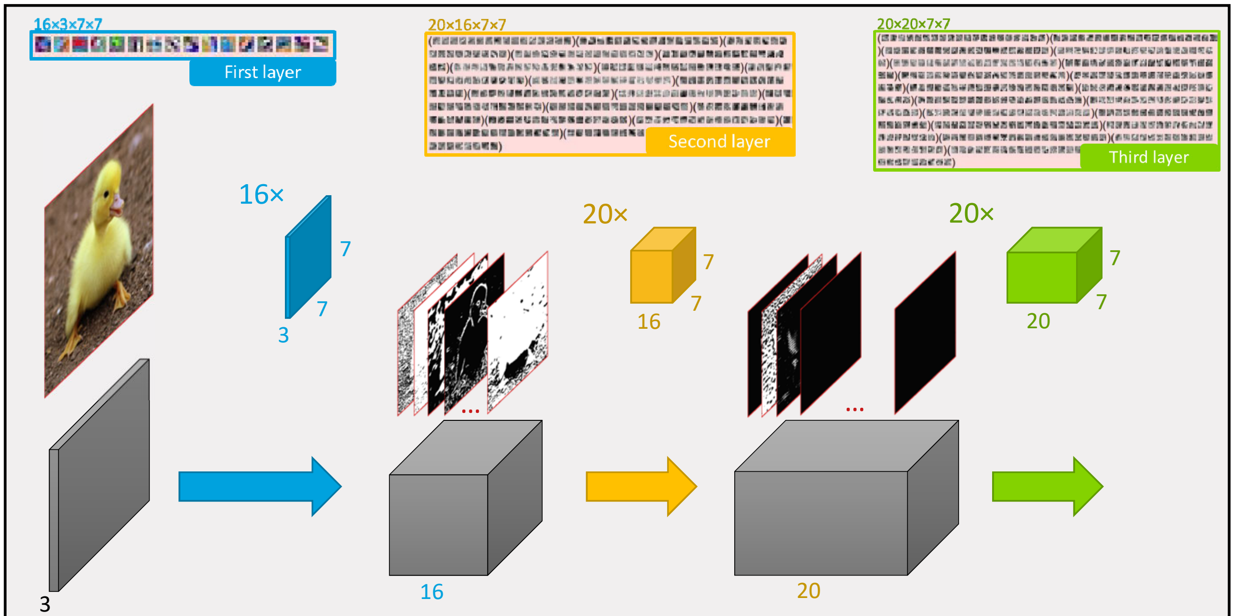
Can't visualize deeper layers as images…

- Example: filter kernels of ConvNetJS

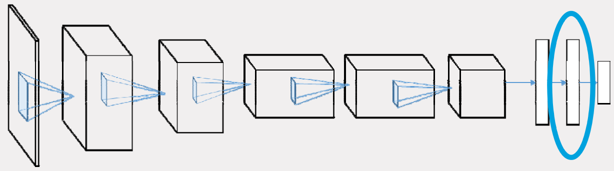
- First layer shows interpretable kernels because they work directly on the image
- The following kernels work on intermediate results, which
  - a) *Typically have more channels than we can visualize*
  - b) *Work on a filtering result of the previous layer, which are hard to interpret (for us)*

Need some additional technique to visualize these layers





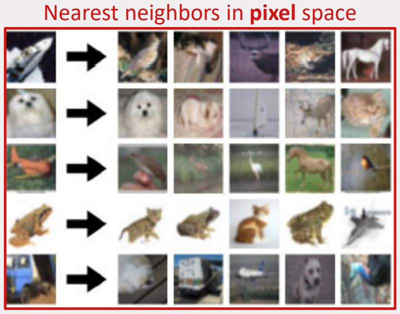
Visualization



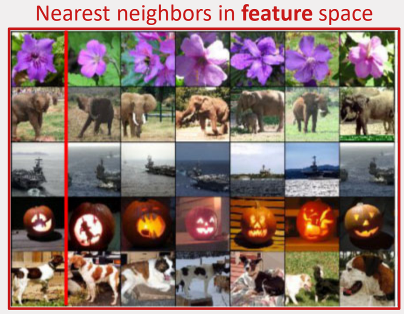
What happens in the last layer?

- Take the output of the last FC layer
- Check where these CNN encodings end up in feature space

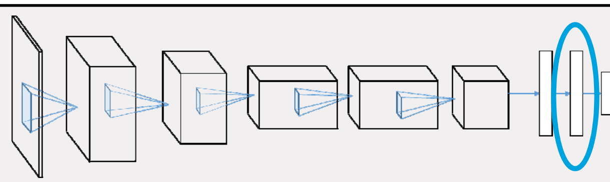
Nearest neighbors in **pixel** space



Nearest neighbors in **feature** space

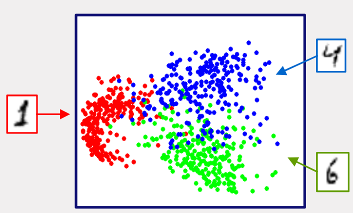


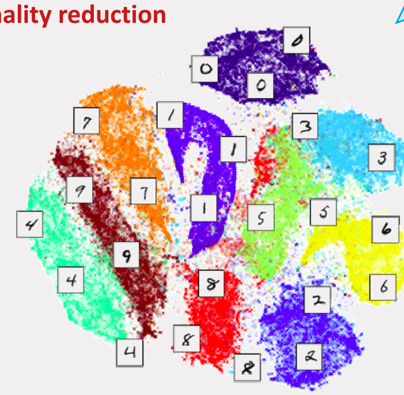
Visualization



What happens in the last layer?

- Alternative to looking at the nearest neighbors: **Dimensionality reduction**
  - *E.g. PCA*

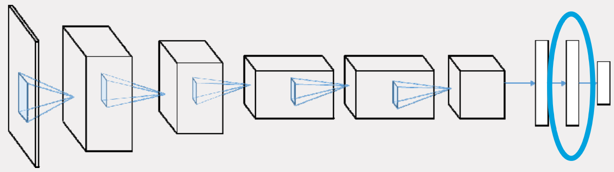




  - *...or something more advanced:*
- **t-Distributed Stochastic Neighbor Embedding (t-SNE)**

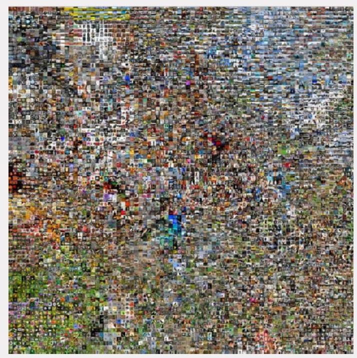
*image:https://bigsnarf.wordpress.com/2016/11/17/t-sne-attack-data/

Visualization



What happens in the last layer?

- Alternative to looking at the nearest neighbors: **Dimensionality reduction**



1. Run images through ConvNet;
2. Map feature vectors to 2D feature space with t-SNE;
3. Visualize where images show up.

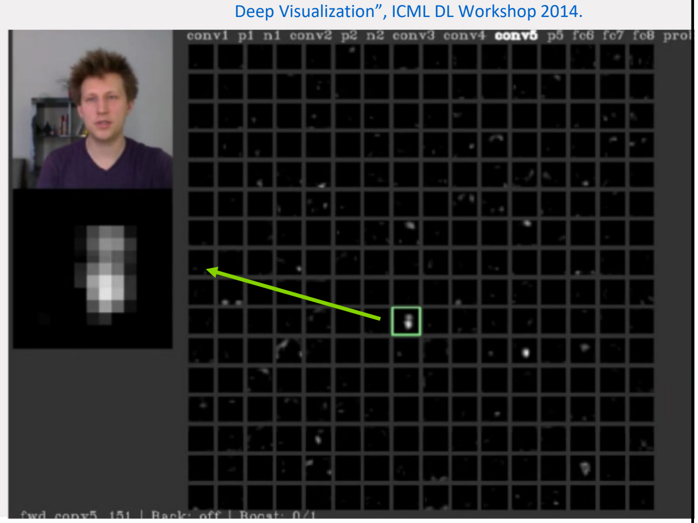
high-resolution versions at http://cs.stanford.edu/people/karpathy/cnnembed/

Visualization

Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

Looking at the activations

- Real-time visualization tool
- Run AlexNet on webcam input and show responses of certain activation layer
- conv5 feature map is 128×13×13
- visualize as 128×13×13 grayscale images

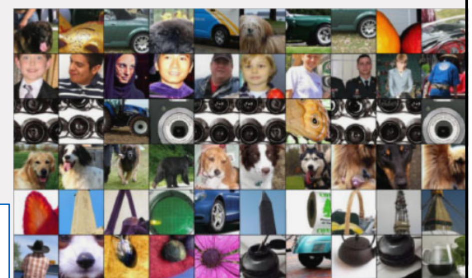


Visualization

Maximally activating patches for different neurons in a conv layer

Maximaly activating patches

- What sort of patches cause maximal activations in different neurons?
- Pick a layer and a channel, run many images through the network and record values of chosen channel
- Visualize image patches that correspond to maximal activations



Q: Why does the deeper layer looks at a different scale?

Maximally activating patches for different neurons in a deeper conv layer

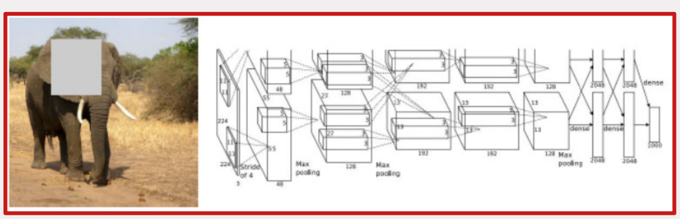
Springenberg et al, “Striving for Simplicity: The All Convolutional Net”, ICLR Workshop 2015

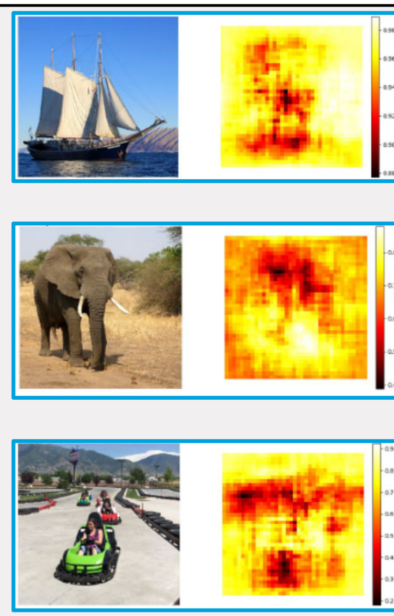
Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

Visualization

Visualization by occlusion

- Which part of the input image are important for classification?
- Use a sliding window to block a certain image part and record predicted probabilities for the image (for the true class)
- Assumption: blocked-out parts that cause a drop in probability are important for the network decision



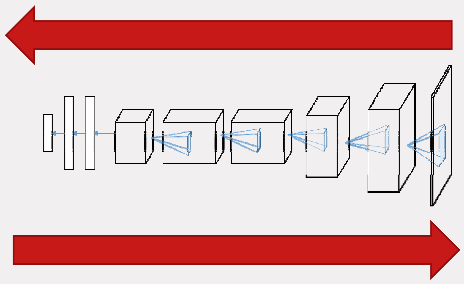


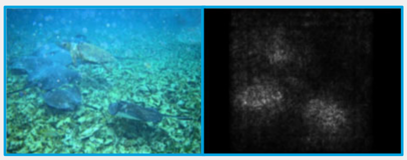
Simonyan, Vedaldi, and Zisserman, "*Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*", ICLR Workshop 2014

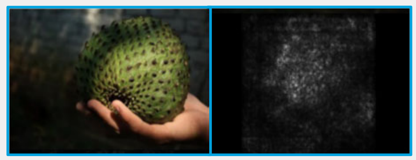
Visualization

Saliency maps

- Backprop gradient of the loss to pixels of the image
- See how much loss certain will inflict by changing value
- Yields an idea for how important certain pixels are for classification



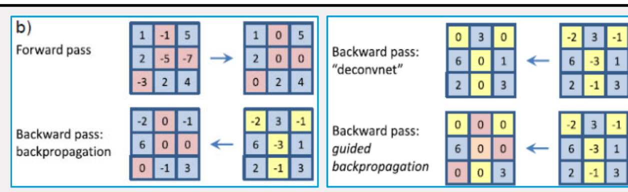


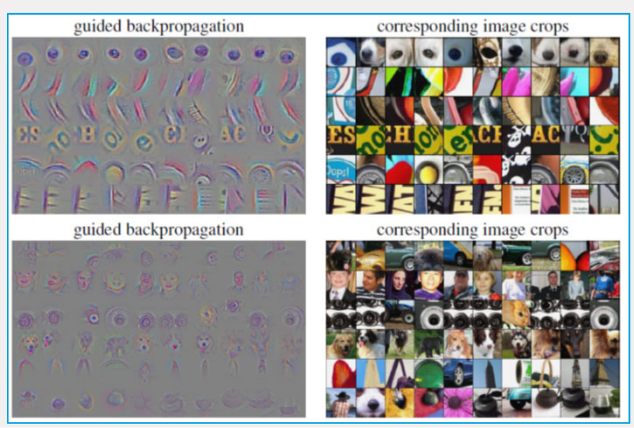


Visualization

Guided backprop to visualize intermediate features

- Which parts of image patches cause the neuron to fire?
- Compute saliency maps for single neurons
- Backprop only positive gradients through each ReLU





Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015

Visualization

Until now: methods all a function of a particular input image

- More interesting: **what type of pattern in general would cause the neuron to fire?**

Gradient ascent for visualizing CNN features

- Fix weights of CNN
- Use gradient ascent to synthesize an image that maximizes the output of a certain neuron
- How does the network know what an image looks like?

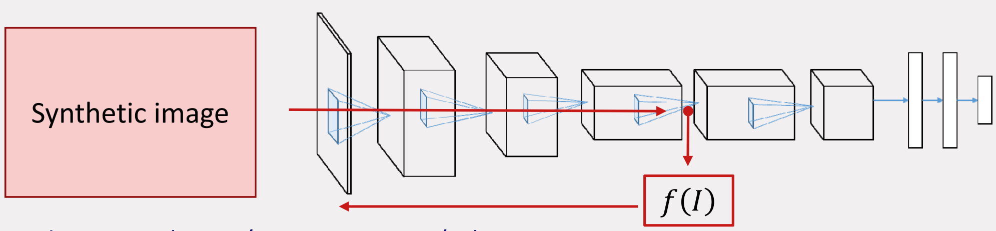
→ Introduce a regularization term to enforce this constraint!

$$I^* = \arg\max_I \underbrace{\boxed{f(I)}}_{\text{Some score or neuron output}} + \underbrace{\boxed{R(I)}}_{\text{Regularization term to enforce natural-looking images}}$$

Visualization

Gradient ascent for visualizing CNN features

$$\boxed{I^* = \arg\max_I f(I) + R(I)}$$

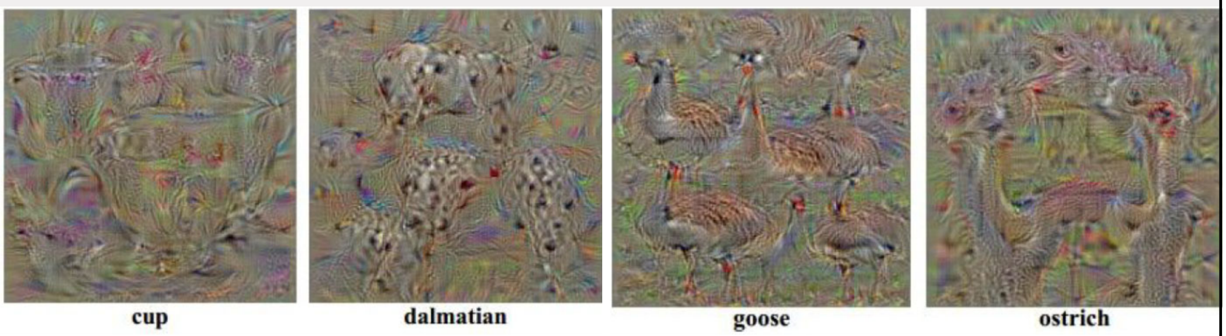


1. Initialize image (zeros / Gaussian noise / …)
2. Forward image to compute current scores
3. Backprop to get gradient of neuron value with respect to image pixels
4. Make a small update to the image

Visualization

Gradient ascent for visualizing CNN features

- Results when L2 regularization is applied
- Note: these are just *possible* images that lead to optimal scores!



Billiard table

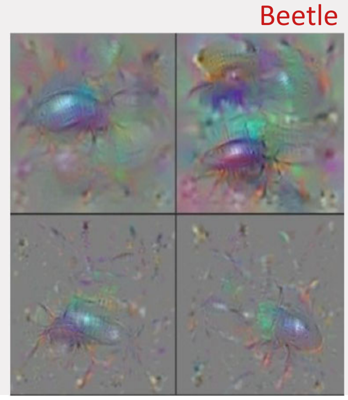
Visualization

Gradient ascent for visualizing CNN features

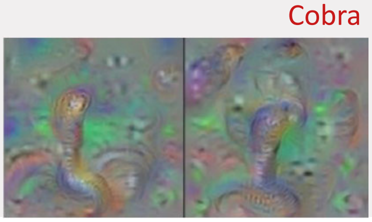
- Apply some tricks to improve the quality of the images
  1) *Apply Gaussian blurring*
  2) *Clip pixels with small values to 0*
  3) *Clip pixels with small gradients to 0*

Flamingo

Beetle



Cobra



Nguyen et al, “*Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks*”, ICML Visualization for Deep Learning Workshop 2016

Visualization

Gradient ascent for visualizing CNN features

- Taking into account the within-class variability leads to even better results

Reconstructions of multiple feature types recognized by the same “grocery” store neuron

Corresponding example training set images recognized by the same neuron

Nguyen et al, “Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks”, ICML Visualization for Deep Learning Workshop 2016

Nguyen A, et al., *"Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images"*, CVPR 2015

Szegedy, Christian, et al. "*Intriguing properties of neural networks*." *arXiv preprint arXiv:1312.6199* (2013).

Goodfellow et al., *"Explaining and Harnessing Adversarial Examples"*, ICLR 2015.

Fooling the network

We can apply the same procedure to find...

examples that maximally fool the network!

➢ Maximize a target class score by applying a small modification to the image

Example from cs231n Lecture 12 slide 36

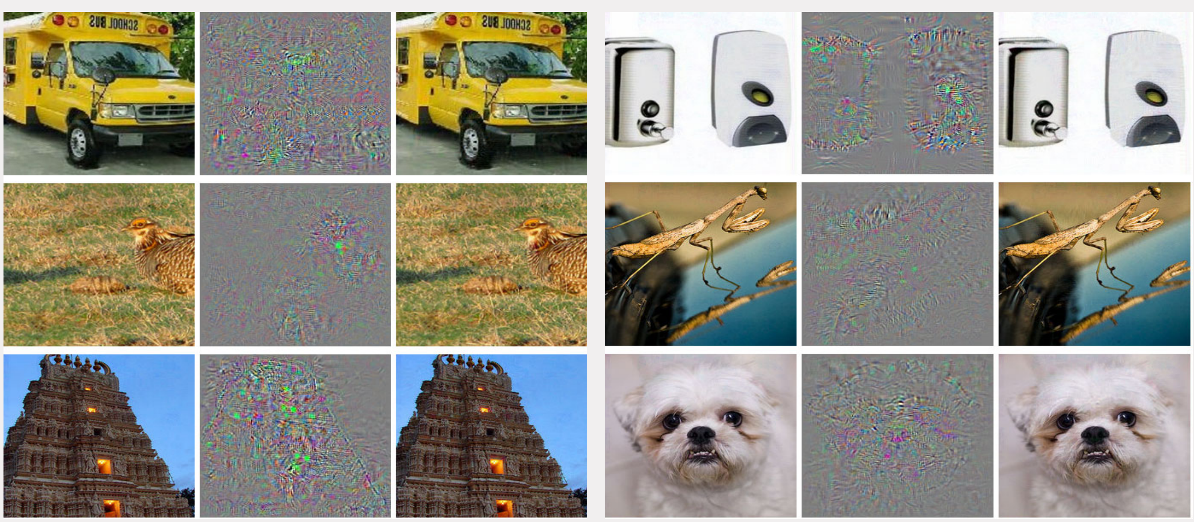


Image | Noise (10x) | Image + noise | Image | Noise (10x) | Image + noise

After (imperceivable) distortion

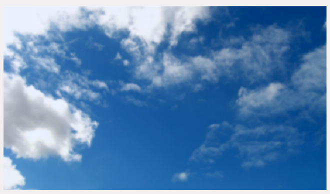
AlexNet prediction = "Ostrich"

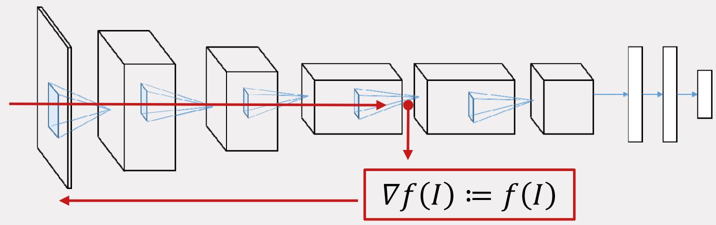
Images: http://goo.gl/huaGPb

Beyond visualization

Deep dream

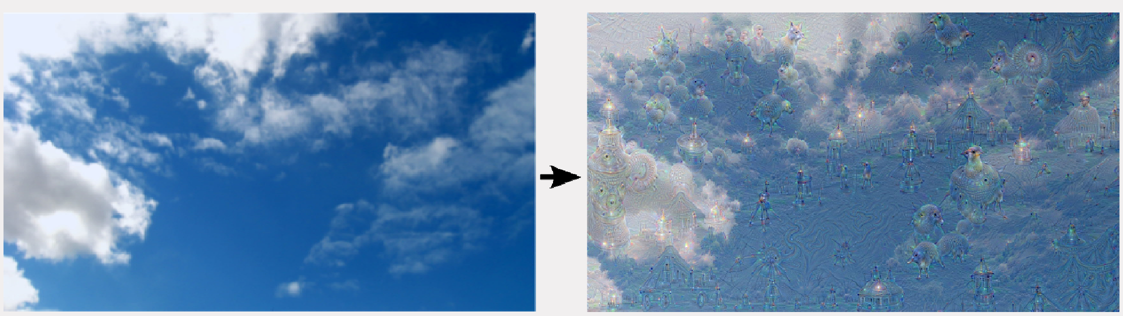
- Visual illusions of the network: “CNN on acid”
- Amplify activations of a certain layer and project them back on the image
  - *Think about e.g. cloud patterns*



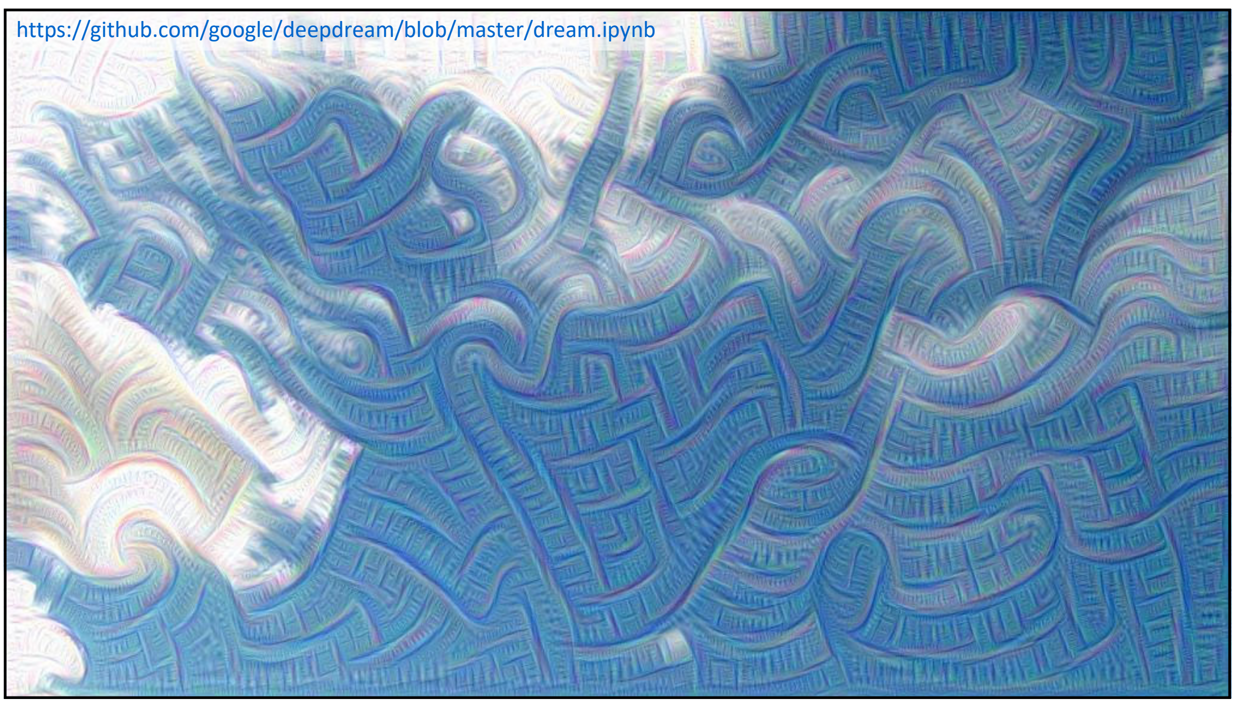


Beyond visualization

Deep dream




https://github.com/google/deepdream/blob/master/dream.ipynb


https://github.com/google/deepdream/blob/master/dream.ipynb

Mahendran and Vedaldi, "*Understanding Deep Image Representations by Inverting Them*", CVPR 2015

Beyond visualization

Feature inversion

- Reconstruct an image from its feature representation at layer point in the network
- Find a new image that has a feature space representation close to some target feature vector
- Make the image look natural → regularize

Φ_0 Given feature vector

$\Phi(\mathbf{x})$ Features of image $\mathbf{x}$

Find image: $$\mathbf{x}^* = \arg\min_{\mathbf{x}} \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda\mathcal{R}(\mathbf{x})$$

With distance: $$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$$

And Regularizer: $$\mathcal{R}_{V^\beta}(\mathbf{x}) = \sum_{i,j}\left(\left(x_{i,j+1} - x_{i,j}\right)^2 + \left(\left(x_{i+1,j} - x_{i,j}\right)\right)^2\right)^{\frac{\beta}{2}}$$

Johnson, Alahi, and Fei-Fei, *"Perceptual Losses for Real-Time Style Transfer and Super-Resolution"*, ECCV 2016

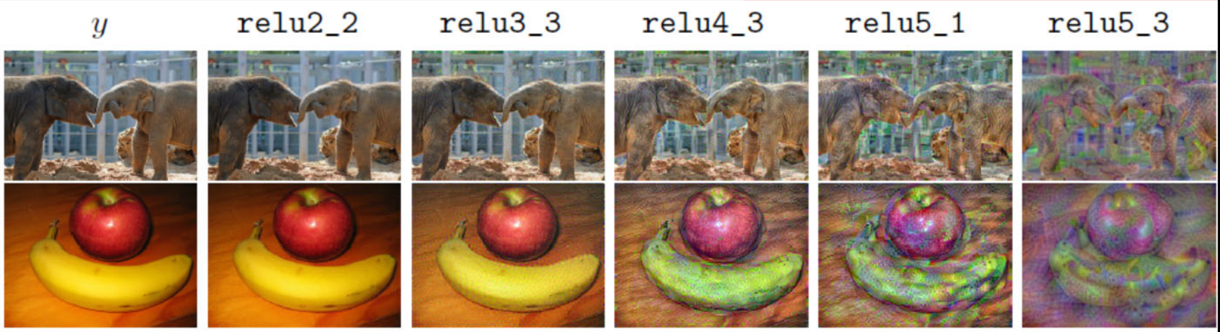
Mahendran and Vedaldi, *"Understanding Deep Image Representations by Inverting Them"*, CVPR 2015

Beyond visualization

Feature inversion

- Low-level information is lost as we go deeper into the network

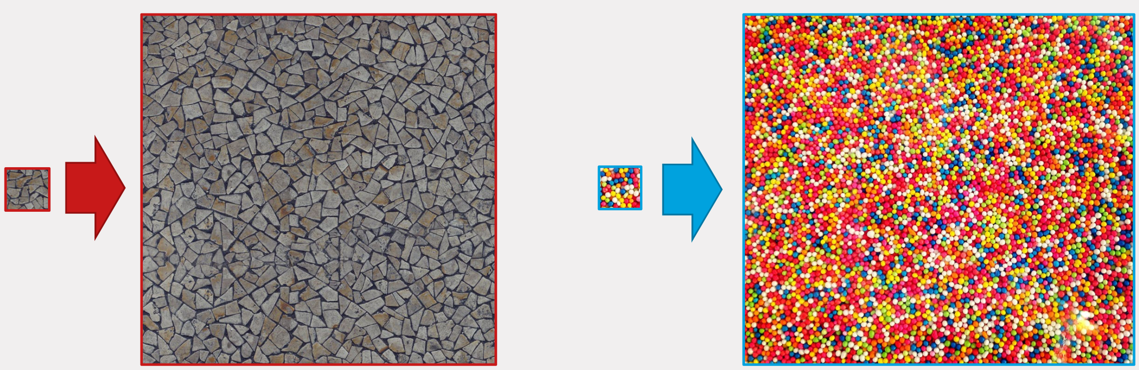
Reconstructing from different layers of VGG-16



Beyond visualization

Texture synthesis

- Given a sample patch of some texture, can we generate a bigger image of the same texture?



Beyond visualization

Texture synthesis

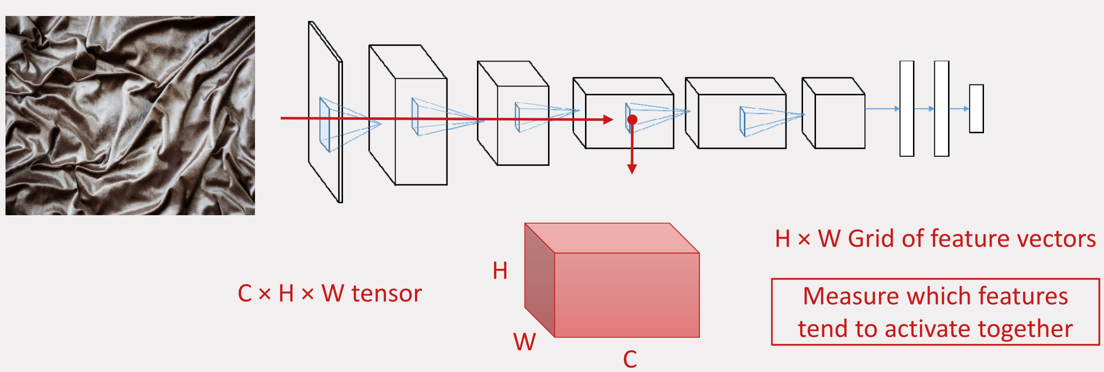
- Given a sample patch of some texture, can we generate a bigger image of the same texture?
- Old problem in computer graphics, see (Wei and Levoy, 2000) or (Efros and Leung, 1999)
  - *Works pretty well for simple textures*
- Classical methods break down for more complex textures

Alternative: Neural texture synthesis

Beyond visualization

Neural texture synthesis

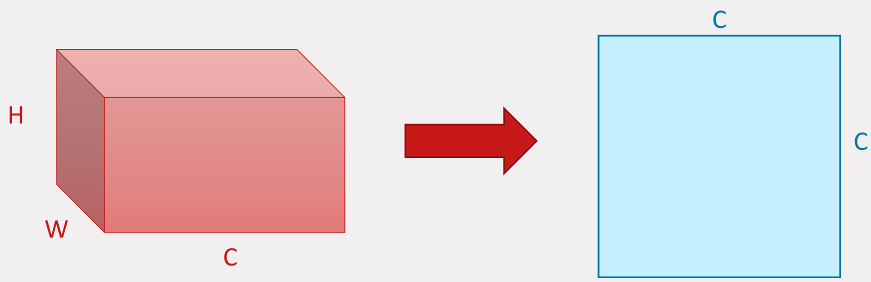
- Feed input texture to CNN and get the convolutional output at some layer in the network



Beyond visualization

Neural texture synthesis

- Measure “co-occurrence” between features by constructing the Gram matrix
  - *Take outer product between feature pair*
  - *Average over all feature pairs to obtain the Gram matrix*



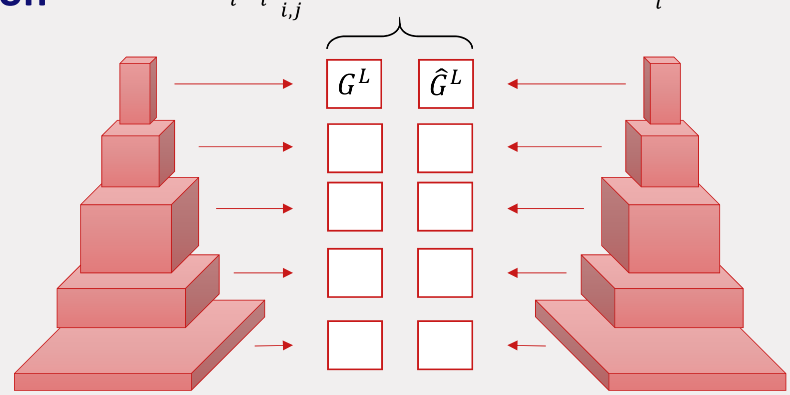
Gram matrix: average over all outer products of all feature pairs

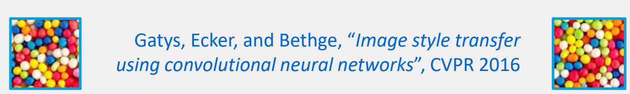
Beyond visualization

$$E_l = \frac{1}{4N_l M_l} \sum_{i,j} \left(G_{i,j}^L - \hat{G}_{i,j}{}^L \right)^2 \qquad \mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \sum_l w_l E_l$$

Neural texture synthesis

- Use a pre-trained model and run on texture image
- Compute Gram Matrices G^L for each of the layer L
- Generate image on a parallel network and enforce similar Gram matrices G^L
- Use backpropagation to iteratively improve result
  - *Same approach as for feature visualization(!)*





Gatys, Ecker, and Bethge, “*Image style transfer using convolutional neural networks*”, CVPR 2016

Beyond visualization

Neural texture synthesis

- Results look pretty good!
- Deeper layers can reconstruct larger patterns from the image

Q: Why?

Gatys, Ecker, and Bethge, "Texture Synthesis Using Convolutional Neural Networks", NIPS 2015

Original

Conv 1

Pool 1

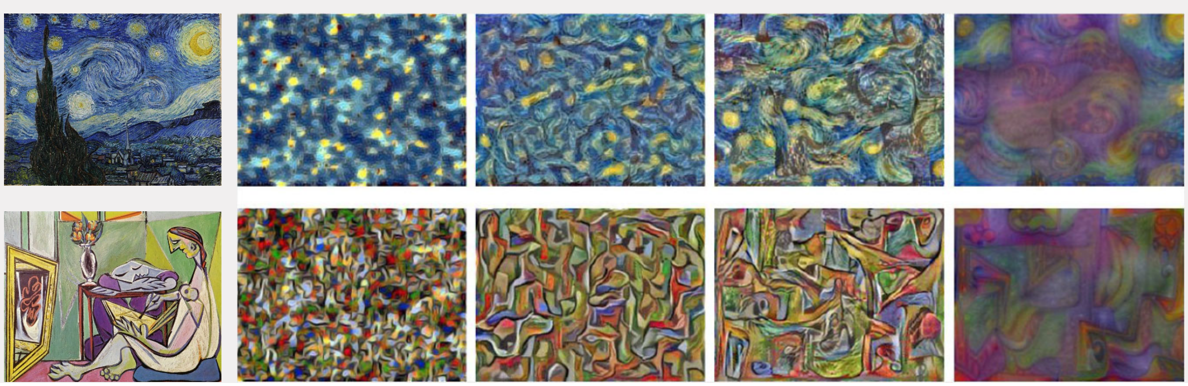
Pool 2

Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016.

Beyond visualization

Neural texture synthesis

- What if we used pieces of art instead?



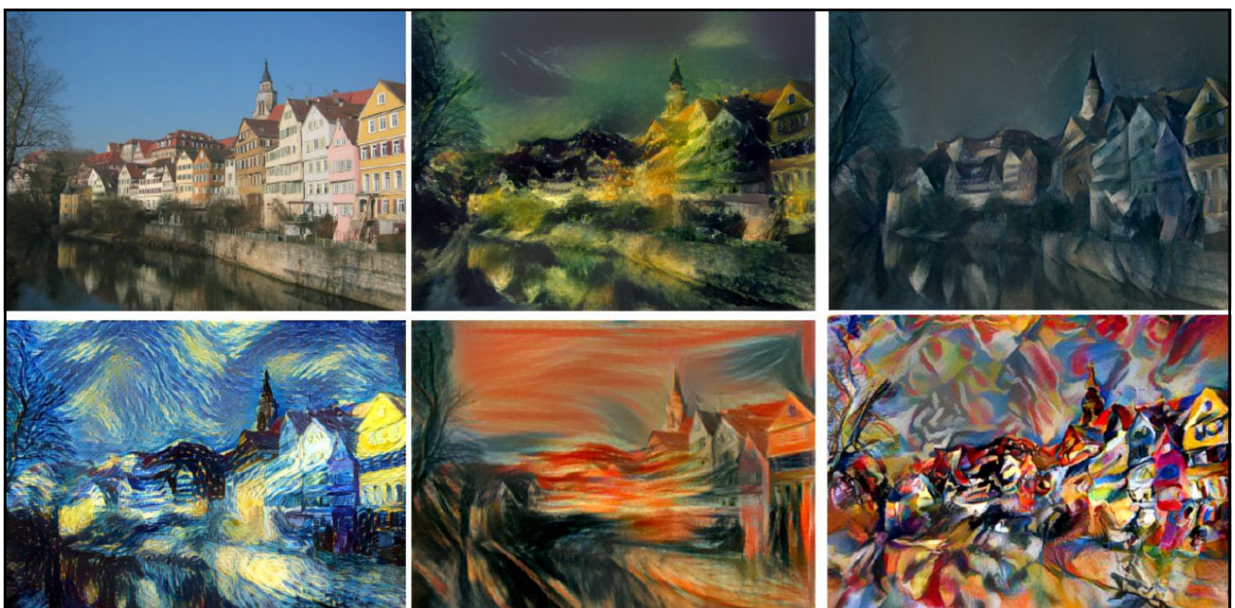
Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016

Beyond visualization

Style transfer

- Combine texture synthesis by Gram matrix matching with feature matching
- Two input images:
  1) *Content image → What configuration of objects do we want?*
  2) *Style image → How do we want the image to look?*

Example from cs231n Lecture 12 slide 67

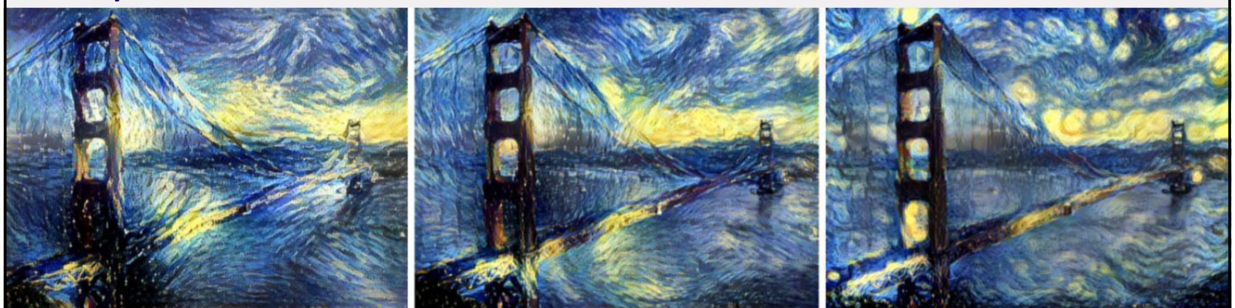


https://github.com/jcjohnson/neural-style

Beyond visualization

Gatys, Ecker, and Bethge, "*Image style transfer using convolutional neural networks*", CVPR 2016

Style transfer



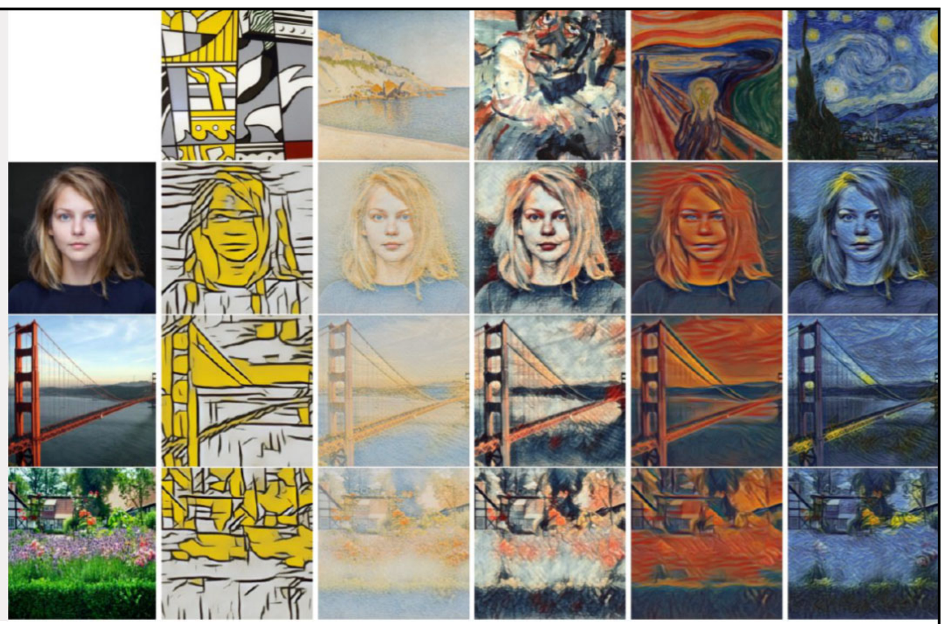
Larger style ⟷ Smaller style

Beyond visualization

Style transfer

- One trained network for many different styles!

Dumoulin, Shlens, and Kudlur, "A Learned Representation for Artistic Style", ICLR 2017



Summary

Looking at activations

- Small scale: How do the filter kernels look?
- Large scale: What happes at the last layer? (apply dimensionality reduction)
- Medium scale: Occlusion sliding window / maximally activating patches

Visualizing the gradients

- Saliency maps to visualize the gradient on the image
- Guided backprop to visualize patterns that activate single neurons

Beyond visualization

- We can use the same techniques to find images that maximally fool CNNs!
- And do other cool stuff like deep dream, texture synthesis and style transfer!