FSAN/ELEG815: Statistical Learning

Gonzalo R. Arce

Department of Electrical and Computer Engineering
University of Delaware

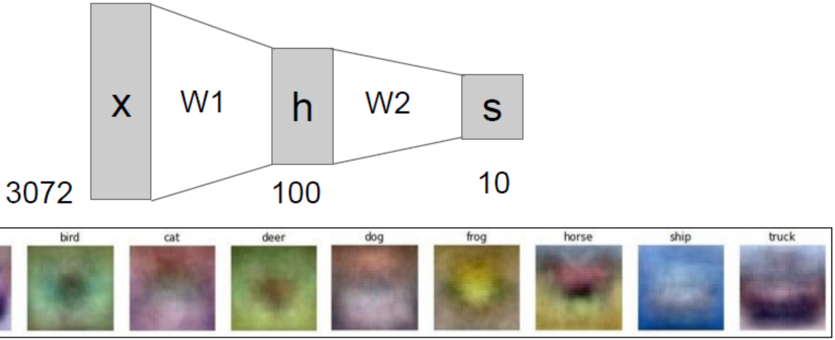
XII: Convolutional Neural Networks

Outline

- Convolutional Neural Networks Overview
- Applications: Style Transfer

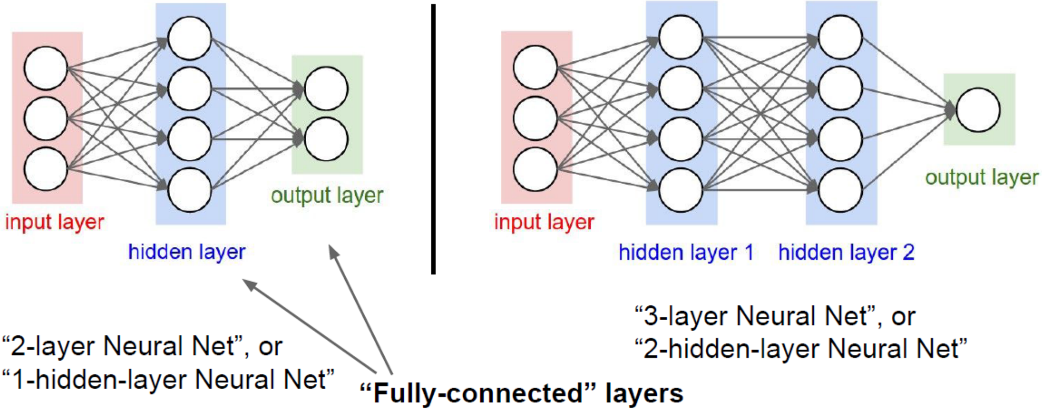
Neural Networks Architectures

- Consider several outputs. Linear score function: $\mathbf{h} = \mathbf{W}\mathbf{x}$
- 2-Layer Neural Network: $\mathbf{s} = \mathbf{W2}\,\theta(\mathbf{W1x})$



Map the raw image pixels to class scores. Classification based on the score.

Neural Networks Architectures

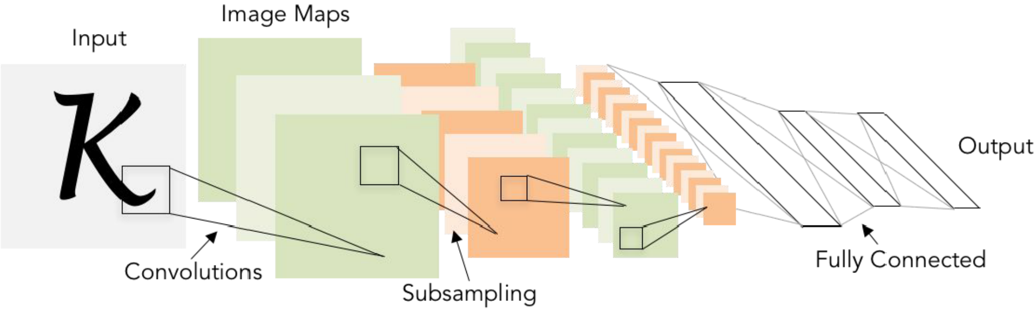


$4+2=6$ neurons.
$[3\times 4]+[4\times 2]=20$ weights
$4+2=6$ biases.

$4+4+1=9$ neurons.
$[3\times 4]+[4\times 4]+[4\times 1]=32$ weights
$4+4+1=9$ biases.

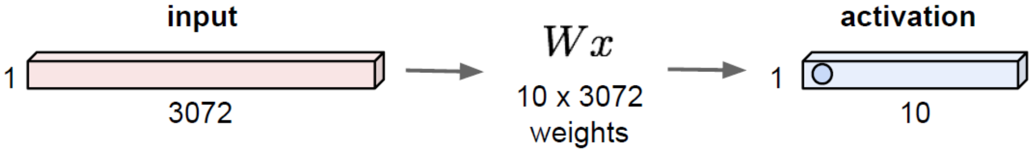
Convolutional Neural Networks Architectures

- Very similar to ordinary Neural Networks.
- Add convolutional layers. Neurons with 3 dimensions: width, height and depth.
- Inputs are also volumes.



Neural Network - Fully Connected (FC) Layer

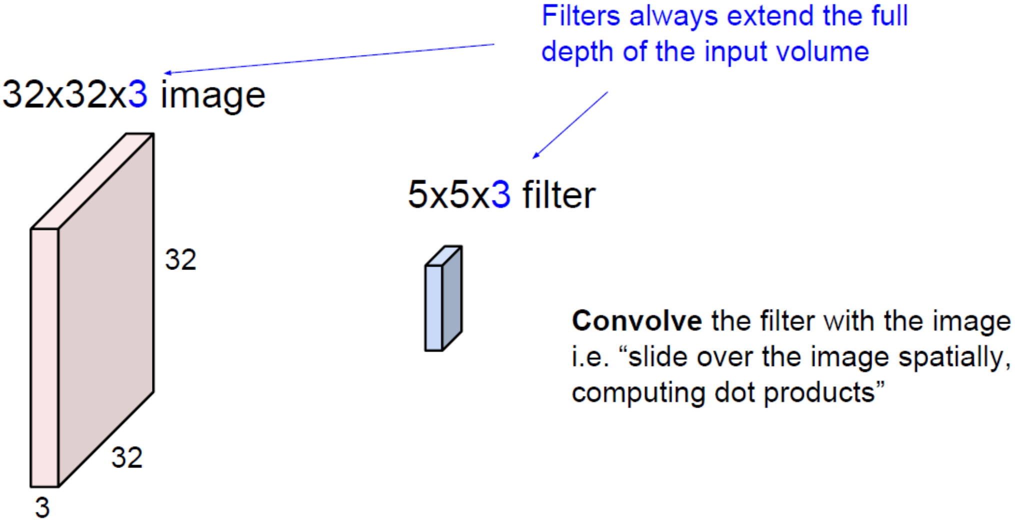
Consider a $32 \times 32 \times 3$ image $\rightarrow$ stretch to 3072×1



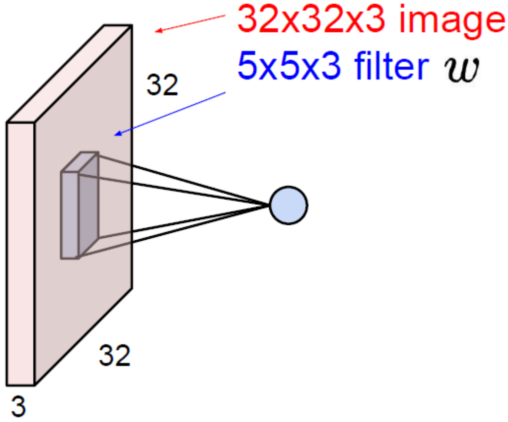
Each output is the result of a dot product between a row of $\mathbf{W}$ and the input $\mathbf{x}$. 10 neurons outputs.

Convolutional Layer

Consider a $32 \times 32 \times 3$ image $\rightarrow$ preserve spatial structure.



Convolutional Layer



Result: dot product between the filter and a small $5 \times 5 \times 3$ chunk of the image.

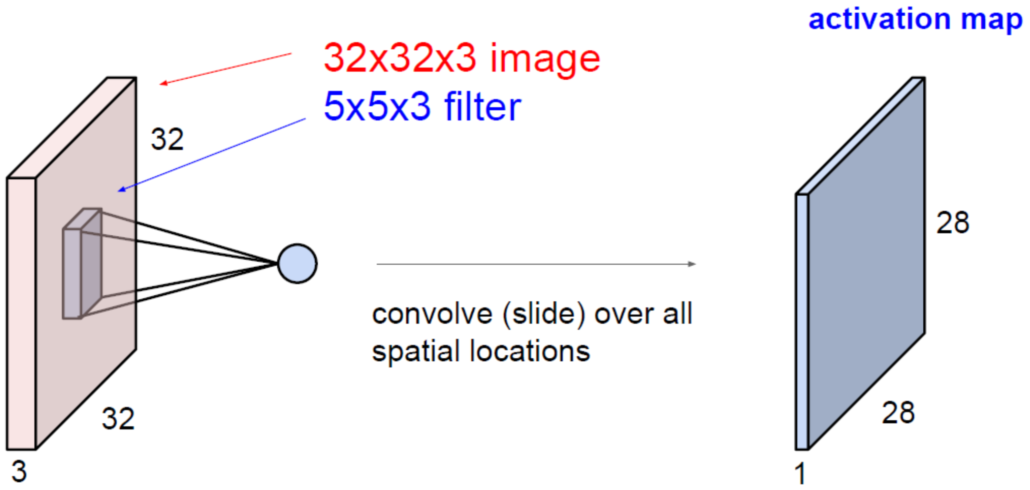
Volume convolution at (x, y), for **all** maps of the input volume:

$$\text{conv}_{x,y} = \sum_i w_i v_i$$

where ws are kernel weights, vs chuck of the image.
Adding scalar bias b:

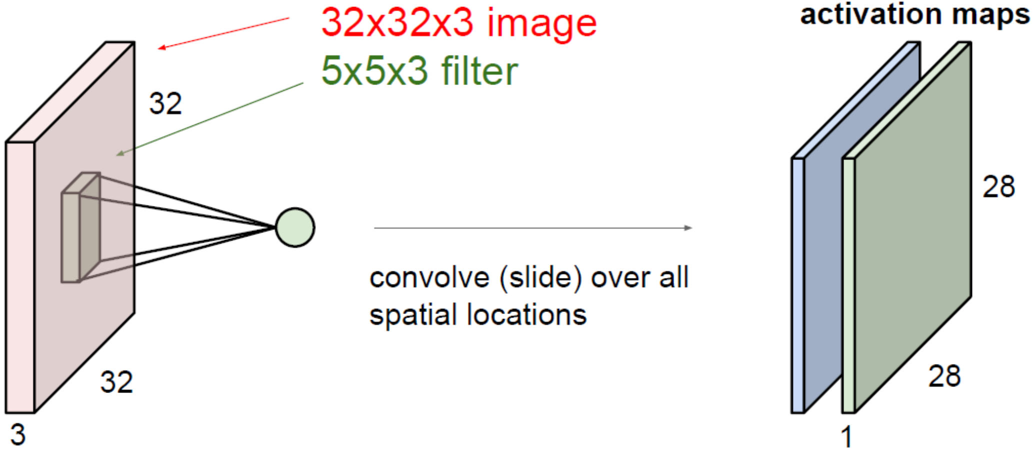
$$z_{x,y} = \sum_i w_i v_i + b$$

Convolutional Layer



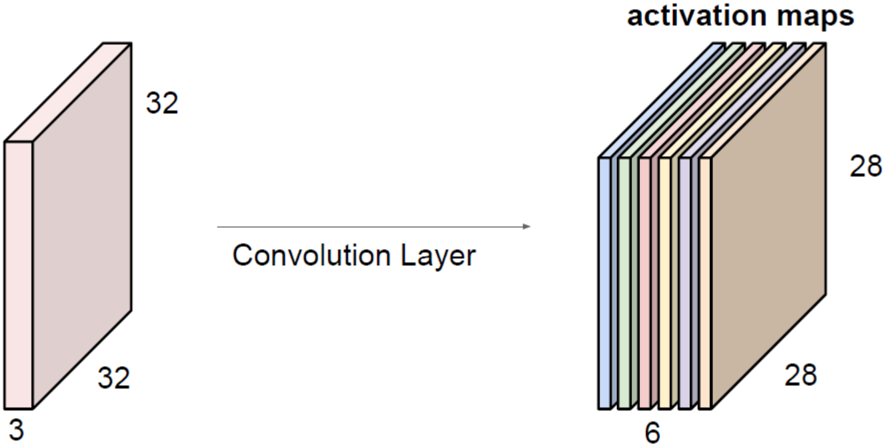
Convolutional Layer

Consider a second, green filter:



Convolutional Layer

Consider 6 filters (5×5), we get 6 separate activation maps:



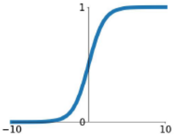
We stack these up to get a "new image volume" of size $28 \times 28 \times 6$

Activation Functions

Pass every element of each activation map through a nonlinearity:

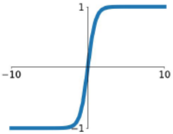
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



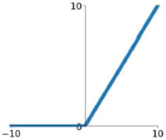
tanh

$\tanh(x)$



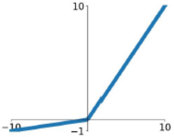
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

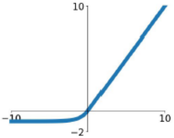


Maxout

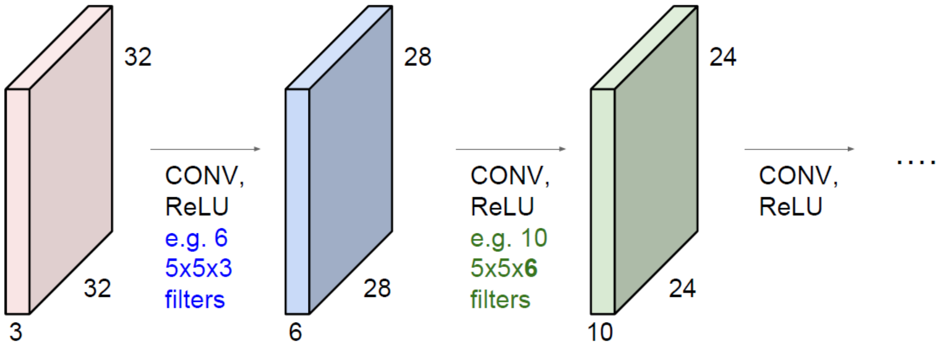
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



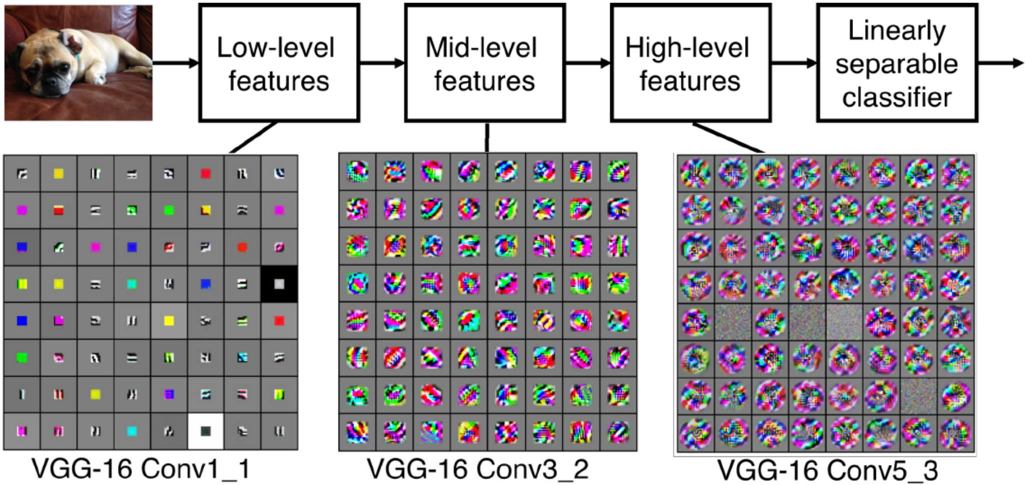
ConvNet is a sequence of Convolutional Layers, interspersed with activation functions:



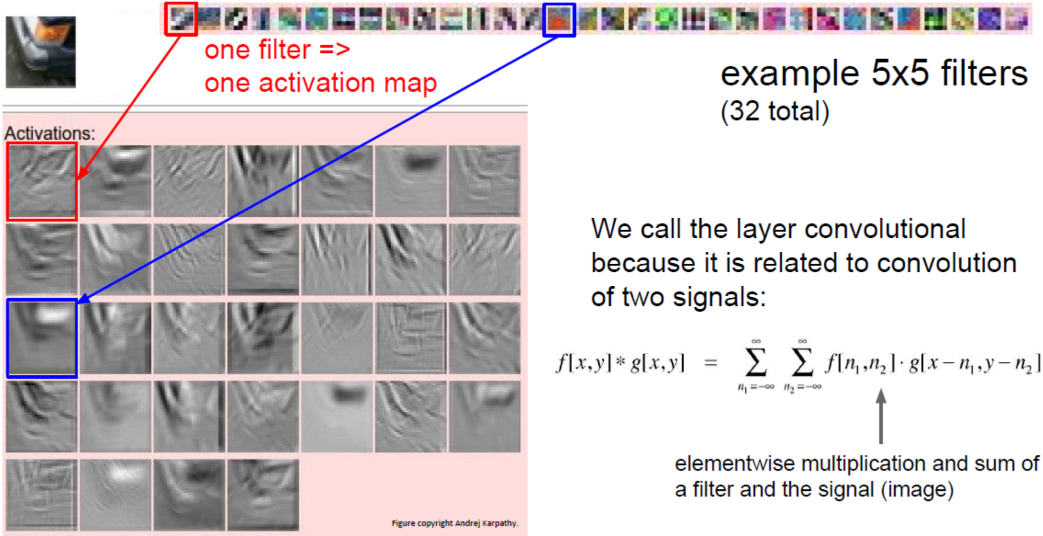
Notice how the activation maps get smaller, this can be solved by zero padding.

Interpretation

Filters Learned:



Interpretation



example 5x5 filters
(32 total)

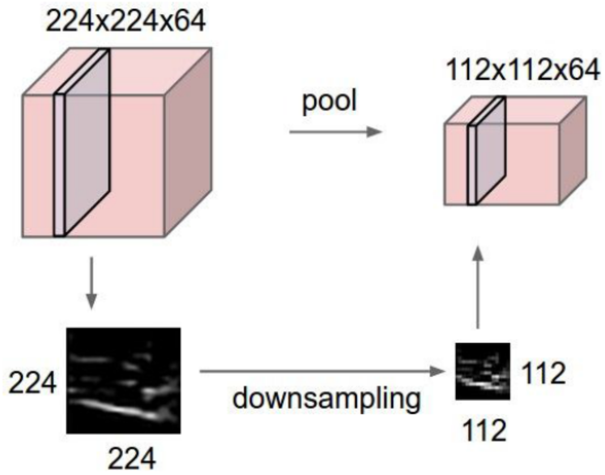
We call the layer convolutional because it is related to convolution of two signals:

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

elementwise multiplication and sum of a filter and the signal (image)

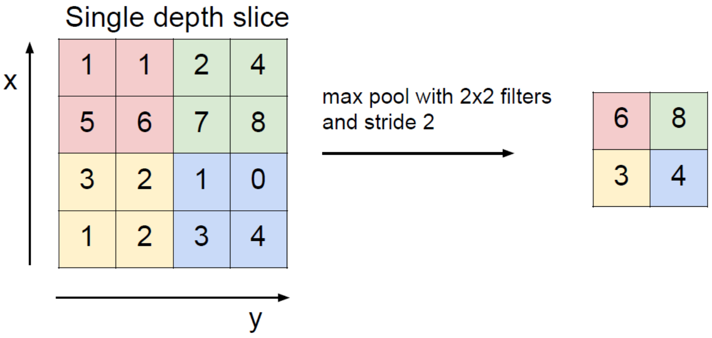
Pooling Layer

- Makes the representations smaller and more manageable.
- Operates over each activation map independently.
- Neighborhood of 2×2 is replaced by the average.



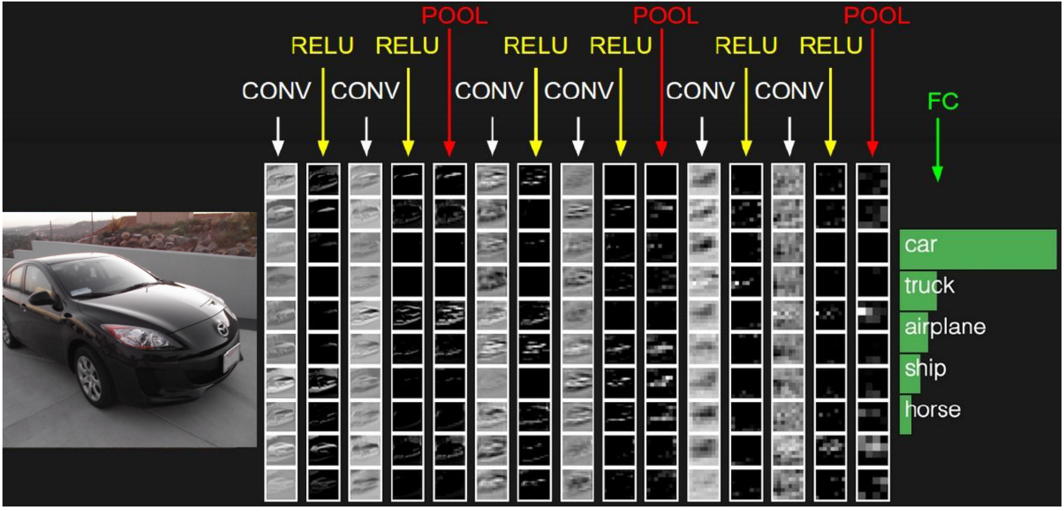
Max Pooling

- Neighborhood of 2×2 is replaced by the maximum value.
- Effective in classifying large image databases.
- Simple and fast.

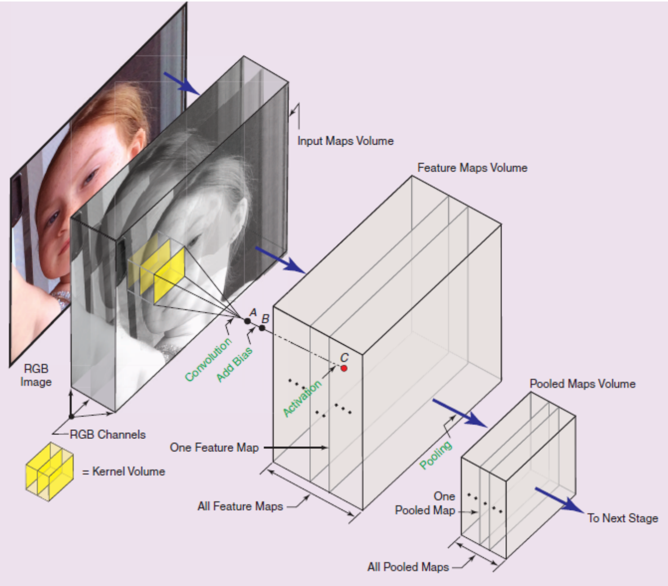


- L_2 pooling is also used. Neighborhood of 2×2 is replaced by the squared root of the sum of their squared values.

Example - Image classification

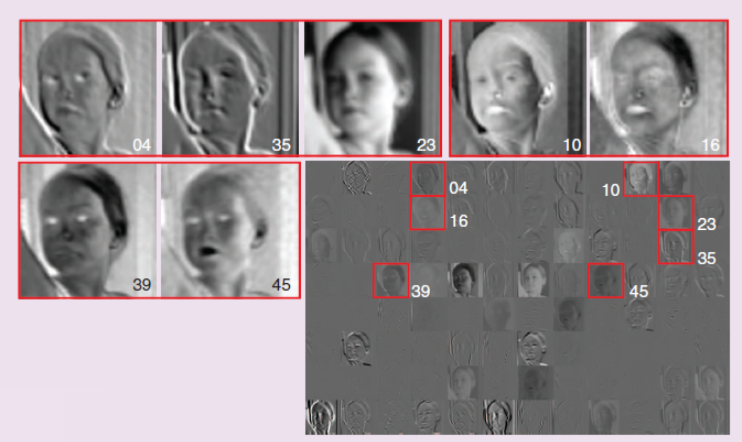


Convolutional Neural Networks Complete Scheme



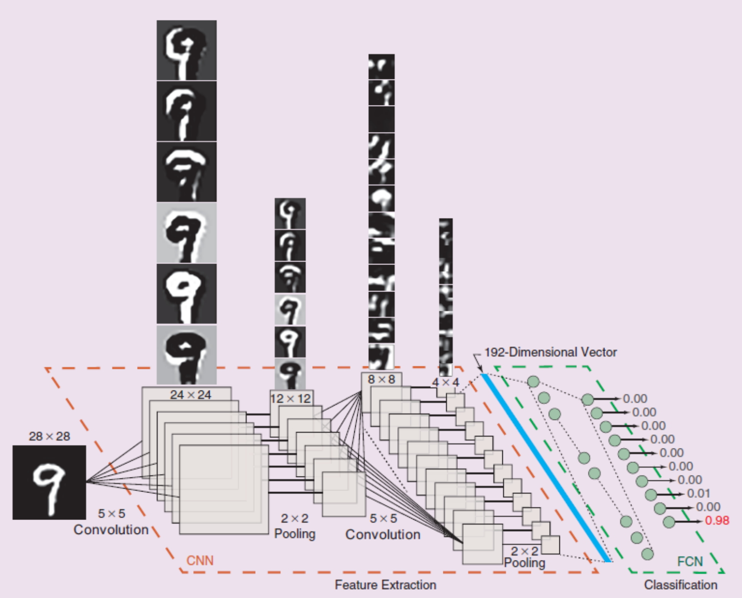
- 277×277 pixels RGB image.
- 96 feature maps.
- 96 kernels volumes of size $11 \times 11 \times 3$
- This weights came from AlexNet: CNN trained using more than 1 million images belonging to 1,000 object categories.

Result Feature Maps



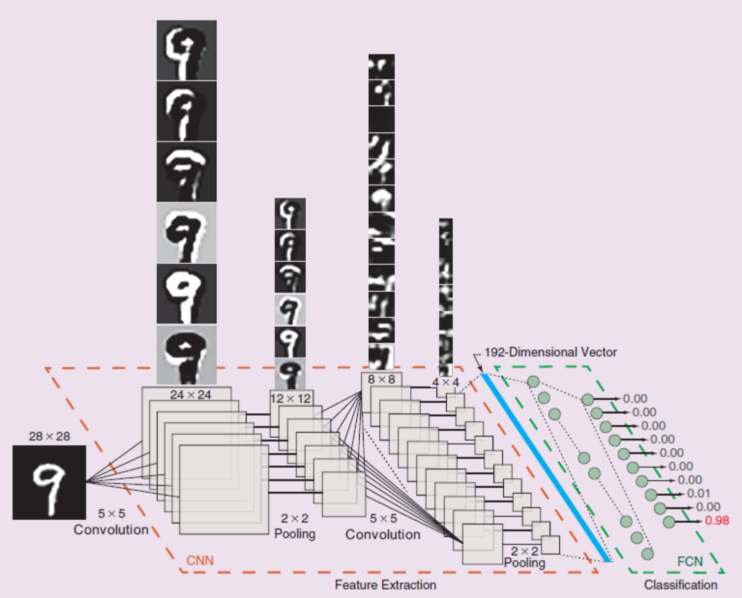
(4) and (35) emphasize edge content. (23) is a blurred version of the input. (10) and (16) capture complementary shades of gray (hair). (39) emphasizes eyes and dress (blue). (45) blue and red tones (lips, hair, skin).

Example - Handwritten Numerals Classification



- Training: 60,000 grayscale images.
- Testing: 10,000 grayscale images.
- Network trained for 200 epochs.
- Performance: 99.4% in training set.
- Performance: 99.1% in testing set.

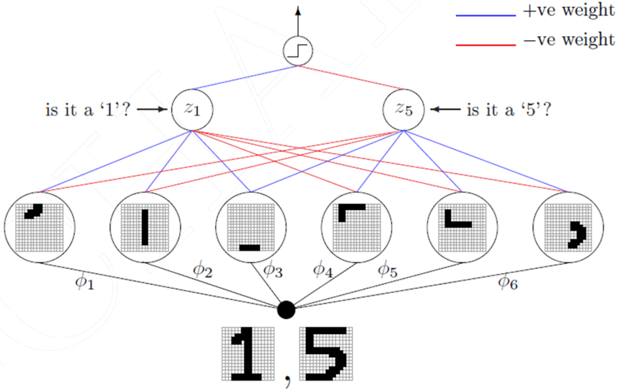
Example - Handwritten Numerals Classification



- First stage: 6 features maps.
- Second stage: 12 features maps.
- Kernels of size 5×5.
- Fully Connected Layer without hidden layers.

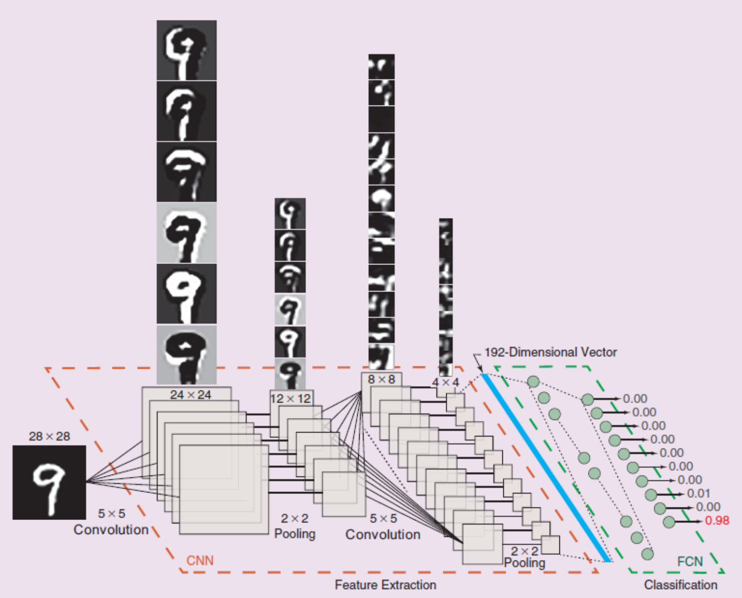
Remember: Networks with many layers - Example

ϕ_i is feature function which computes the presence $(+1)$ and absence (-1) of the corresponding feature.



If we feed in '1', ϕ_1,ϕ_2, ϕ_3 compute $+1$ and ϕ_4,ϕ_5, ϕ_6 compute -1. Combining with the signs of the weights, z_1 will be positive and z_5 will be negative.

Features Map Interpretation

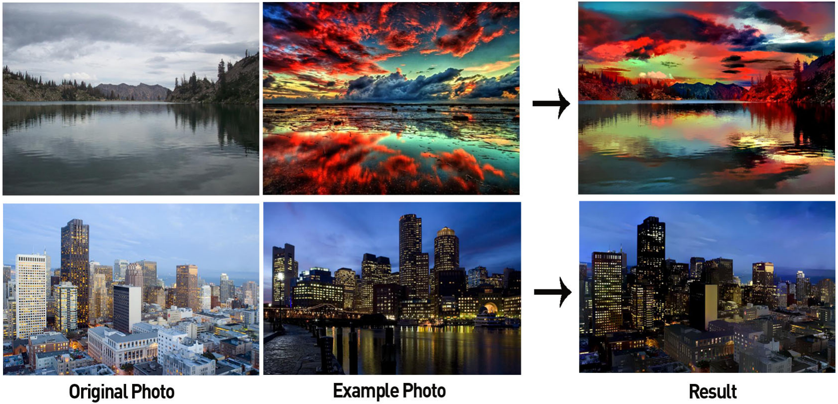


- First feature map: strong vertical components on the left.
- Second: strong components in the northwest area of the top of the character and the left vertical lower area.
- Third: strong horizontal components.

Style Transfer

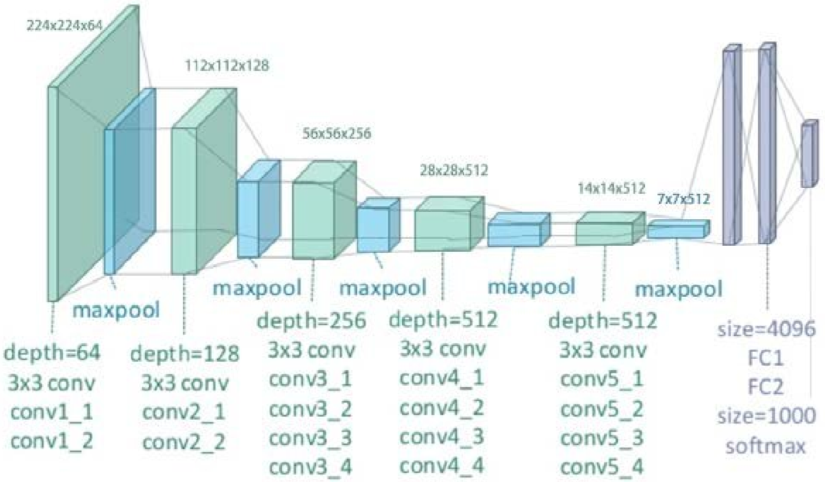
- Goal: Rendering the semantic content of an image in different styles.
- Challenge: separate image content from style.

A Neural Algorithm of Artistic Style can separate and recombine the image content and style of natural images.



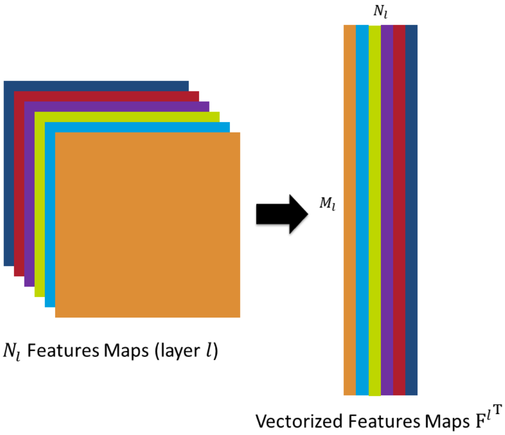
Deep Image Representations

VGG-19 is a convolutional neural network that is trained on more than a million images from the ImageNet database to perform object recognition (1000 categories) and localization.



Content Representation

Responses in a layer l are stored in a matrix $\mathcal{F}^l \in \mathbb{R}^{N_l \times M_l}$ where N_l is the number of filters and M_l is the height times the width of the feature map.



F^l_{ij} is the activation of the i^{th} filter at position j in layer l.

Visualize Image Information at each Layer

Perform gradient descent on a white noise image to obtain a reconstructed image $\vec{x}$ with the information encoded at different layers.
Minimize the loss function:

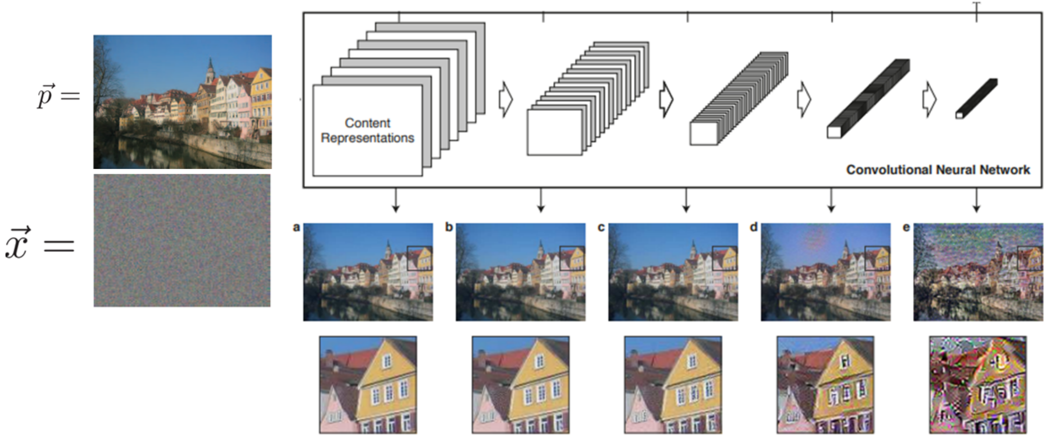
$$\mathcal{L}_{\text{content}}(\vec{p}, \vec{x}, l) = \frac{1}{2}\sum_{i,j}(F_{ij}^{l} - P_{ij}^{l})^2$$

where F_{ij}^{l} and P_{ij}^{l} are the feature representations of the original image $\vec{p}$ and the reconstructed image $\vec{x}$ in layer l.

$$\vec{x}(t+1) = \vec{x}(t) - \lambda\frac{\partial\mathcal{L}_{\text{content}}}{\partial\vec{x}}$$

The gradient with respect to the image $\vec{x}$ can be computed using standard error back-propagation.

Content Representation Results



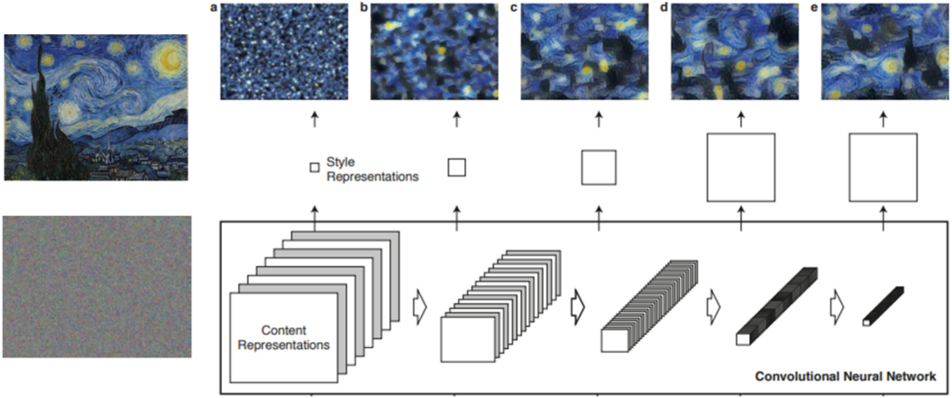
Reconstruction of the input image from layers (a) conv1_2 (b) conv2_2 (c) conv3_2 (d) conv4_2 (e) conv5_2 of the original VGG-Network.

Style Representation

Use a feature space designed to capture texture information: correlation between the different filter responses.

$\vec{a} =$

$\vec{x} =$



Style Representation

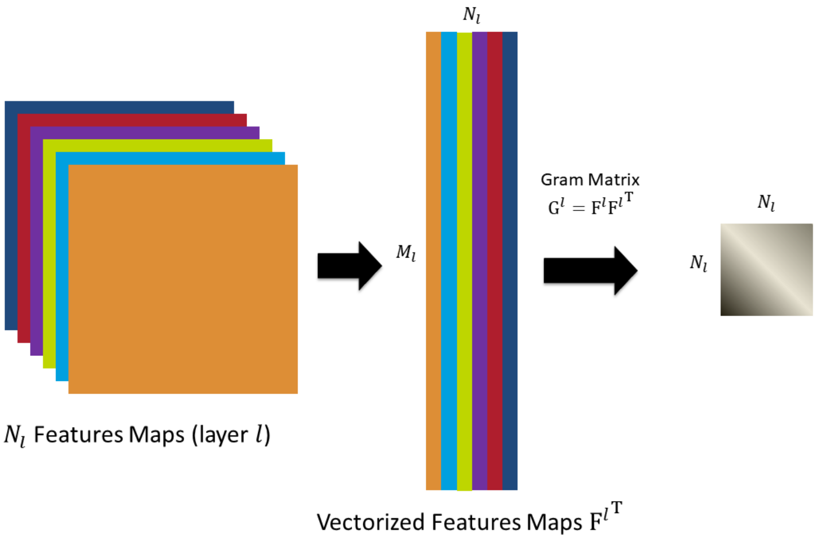
Feature correlations are given by the Gram matrix $G^l \in \mathbb{R}^{N_l \times N_l}$. Expectation taken over the spatial extent of the features maps.

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}$$

G^l_{ij} is the inner product between the vectorized feature maps i and j in layer l.

Perform gradient descent on a white noise image to observe the information captured by these style feature spaces.

Style Representation



Visualize Image Style at each Layer

- Minimize the distance between the Gram matrices. The contribution of layer l to the total loss is:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2$$

where G_{ij}^l and A_{ij}^l are the style representation of the original image $\vec{a}$ and the generated image $\vec{x}$ in layer l.

- The total loss is:

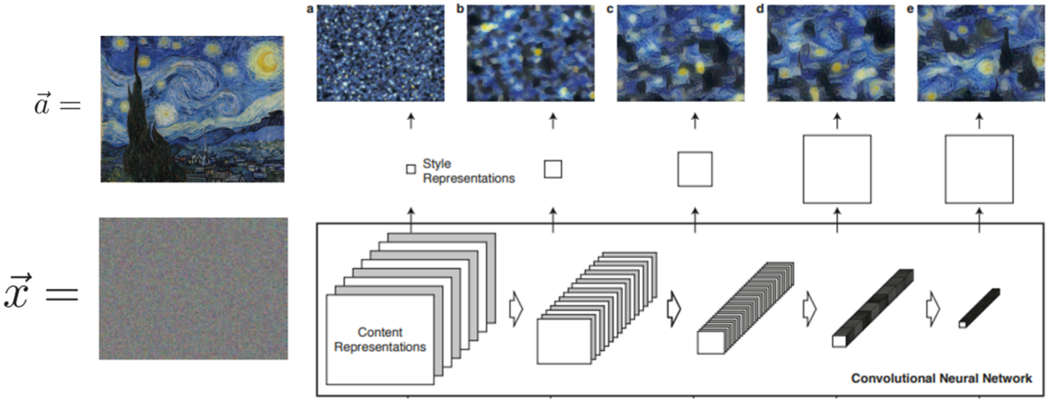
$$\mathcal{L}_{\text{style}}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

w_l are weighting factors (parameters).

$$\vec{x}(t+1) = \vec{x}(t) - \lambda \frac{\partial \mathcal{L}_{\text{style}}}{\partial \vec{x}}$$

The gradient with respect to the image $\vec{x}$ can be computed using standard error back-propagation.

Style Representation Results



Style Reconstructions from layers (a) conv1_1, (b) conv1_1 and conv2_1 (c) conv1_1, conv2_1 and conv3_1 (d) conv1_1, conv2_1, conv3_1 and conv4_1 (e) conv1_1, conv2_1, conv3_1, conv4_1 and conv5_1 of the original VGG-Network.

Style Transfer

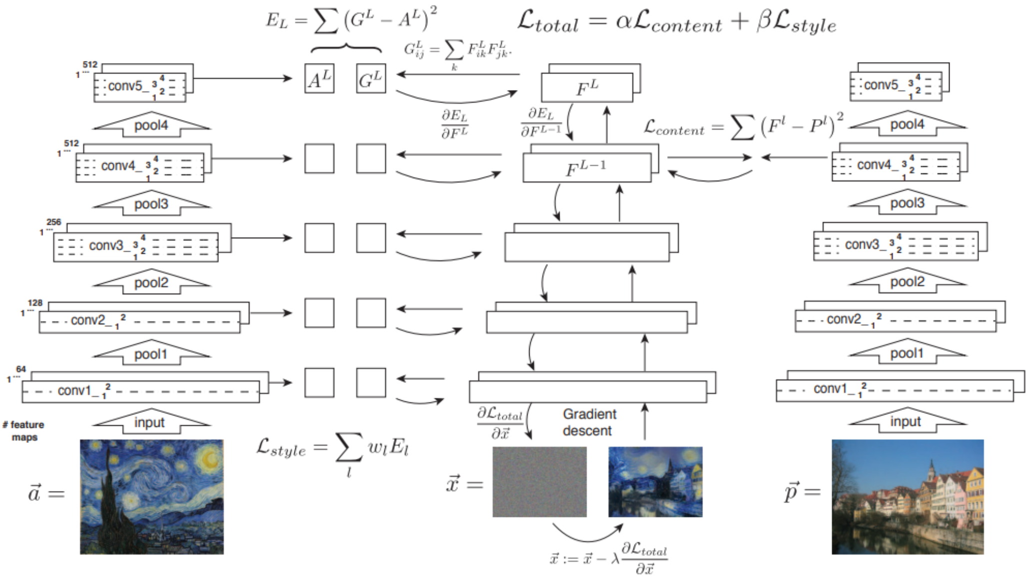
The total loss is a linear combination between the content and the style loss:

$$\mathcal{L}_{\text{total}} = \alpha \mathcal{L}_{\text{content}} + \beta \mathcal{L}_{\text{style}}$$

Its derivative with respect to the pixel values can be computed using error back-propagation.

$$\vec{x}(t+1) = \vec{x}(t) - \lambda \frac{\partial \mathcal{L}_{\text{total}}}{\partial \vec{x}}$$

Style Transfer

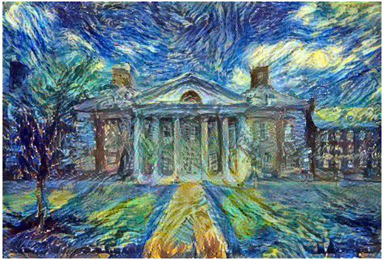


Results

Results

Style Image

Content Image



Results

Style Image

Content Image