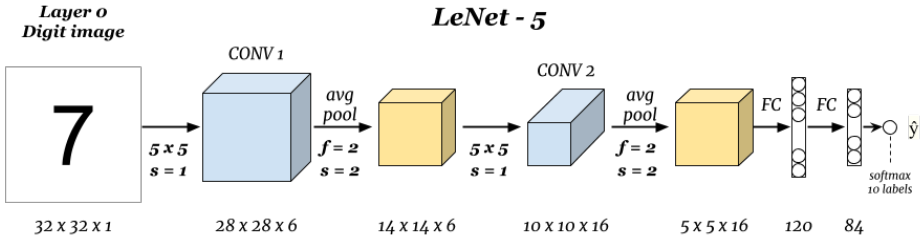
Learning with CNNs

Tianxiang (Adam) Gao

October 14, 2024

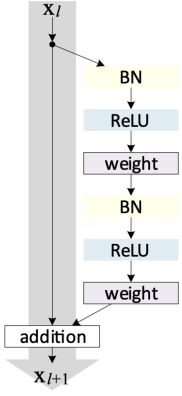
Outline

Recap: Convolutional Neural Networks (CNNs)



- **Convolution operation:** It slides a small **filter** over the input image, performing a **locally linear transformation** to produce a feature map that detects patterns.
- **Padding and stride:** Methods for controlling feature map size, preserving spatial dimensions, and improving *computational efficiency*.
- **Convolution over volumes and with multiple filters:** The input can be multi-channel, and so as the output with the use of multiple filters.
- **Weight Sharing and Sparsity:** Neurons in CNNs **share** weights across locations, with each output relying on a **small, localized** input region.
- **Hierarchical Feature Detection:** Early layers capture basic features (like edges), which later layers combine into higher-level features.

Recap: Stabilizing CNN Training

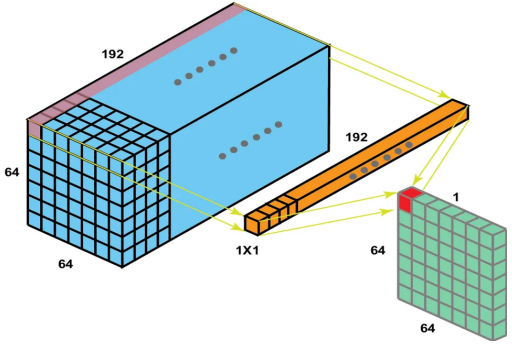


- **Pooling**: Reduces the spatial dimensions of feature maps by downsampling, typically using max or average pooling.
- **Batch Normalization**: Normalizes pre-activation values at hidden layers, mitigating internal covariate shift and accelerating training.
- **Skip Connections**: Create shortcuts by directly adding input to output, stabilizing information flow in deep neural networks.
- **Classic CNNs**: Spatial dimensions shrink while the number of channels increases as depth grows. Overparameterized and deeper CNNs are generally preferred.

Outline

1×1 Convolutions

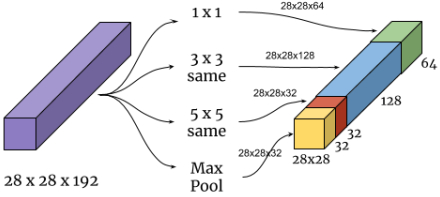
$$\underbrace{\begin{bmatrix} 3 & 0 & 1 \\ 1 & 5 & 8 \\ 2 & 7 & 2 \end{bmatrix}}_{\text{input image } 3 \times 3} * \underbrace{\begin{bmatrix} 2 \end{bmatrix}}_{\text{filter } 1 \times 1} = \underbrace{\begin{bmatrix} 6 & 0 & 2 \\ 2 & 10 & 16 \\ 4 & 14 & 4 \end{bmatrix}}_{\text{feature map } 3 \times 3}$$

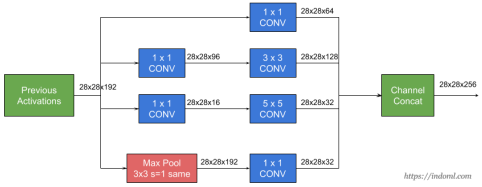


- Recall that each channel in the input data serves as an **indicator map**.
- Each pixel, with multiple channels, represents a set of indicators that capture the distribution of **multiple patterns** within a local region of the original image.
- By applying a 1×1 convolution, we can detect complex patterns that integrate multiple underlying features.
- This approach is useful for combining lower-level features into higher-level representations.

Lin et al. *"Network in Network"*, ICLR 2014.
Image is adapted from here.

Inception Networks

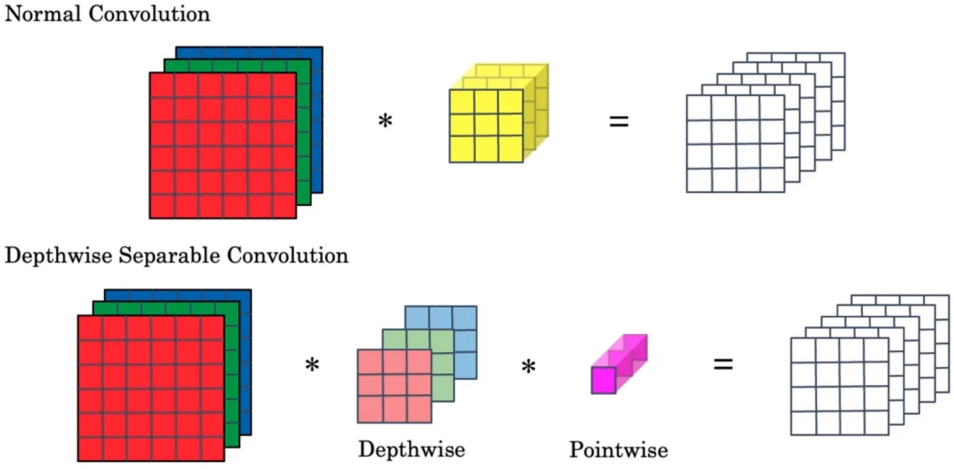




- Inception networks allow the model to select the best filter sizes within a layer, rather than manually choosing them, by providing multiple filter options.
- This approach can be computationally expensive (e.g., a 5x5 filter requires about 120 million operations).
- Applying 1×1 convolutions reduces computation by approximately 90%.

Szegedy et al. *"Going deeper with convolutions"*, CVPR 2014.
Image is adapted from https://indoml.com/.

Mobilenets

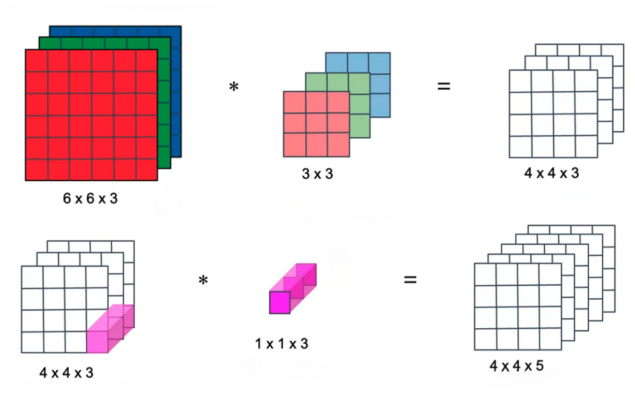


The computational cost of a normal convolution is:

$$\underbrace{2,160}_{\text{Computational cost}} = \underbrace{3 \times 3 \times 3}_{\text{\# params}} \times \underbrace{4 \times 4}_{\text{\# locations}} \times \underbrace{5}_{\text{\# filters}}$$

Howard, Andrew G. *"Mobilenets: Efficient convolutional neural networks for mobile vision applications."* arXiv 2017.

Depthwise Separable Convolution

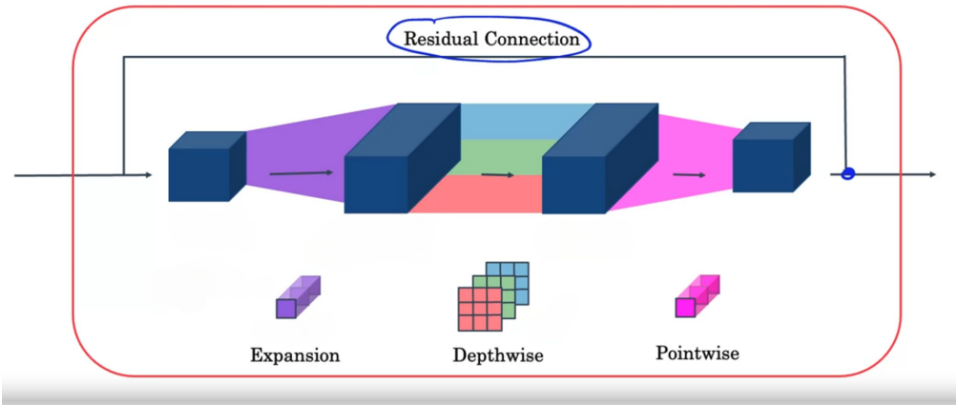


- Each channel has an **independent** filter for the convolution operation.
- The result is then processed using a pointwise or 1×1 convolution.
- Computational cost ratio: $\frac{1}{C_{out}} + \frac{1}{f^2}$

$$672 = \underbrace{3 \times 3}_{\text{\# params}} \times \underbrace{4 \times 4}_{\text{\# locations}} \times \underbrace{3}_{\text{\# filters}} + \underbrace{3}_{\text{\# params}} \times \underbrace{4 \times 4}_{\text{\# locations}} \times \underbrace{5}_{\text{\# filters}}$$

Images adapted from *CS230 Deep Learning* at Stanford.

MobileNetV2



Model sizes:

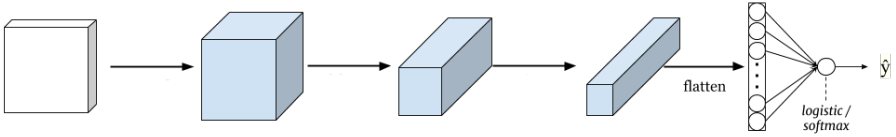
- VGG-16 $\approx$ 528 MB, ResNet-152 $\approx$ 230 MB, Inception $\approx$ 85 MB.
- MobileNet (V1-V4): typically between 5-16 MB.

Sandler, Mark, et al. *"Mobilenetv2: Inverted residuals and linear bottlenecks."* CVPR 2018.

Outline

Transfer Learning

- **Definition**: Transfer learning leverages a **pre-trained** model on a large dataset to solve a new, related task on a smaller dataset.
- **Motivation**: Training deep networks from scratch demands significant data and computational resources. The initial layers in CNNs capture fundamental features that are transferable across various tasks, allowing the higher layers to focus on learning task-specific features.



- **Typical Workflow**:
  1. **Pre-training**: A model is trained on a large dataset (e.g., ImageNet).
  2. **Freezing Layers**: Early layers are frozen to retain their learned features.
  3. **Fine-tuning**: With a lower learning rate, higher layers are retrained on the target dataset, allowing for adaptation to new, task-specific features.
- Frozen early layers can be considered as **fixed** feature extractors, allowing activations to be precomputed, which reduces computation and speeds up training.
- For **larger** target datasets, freeze fewer layers to enable more flexibility in learning higher-level task-specific features during fine-tuning.

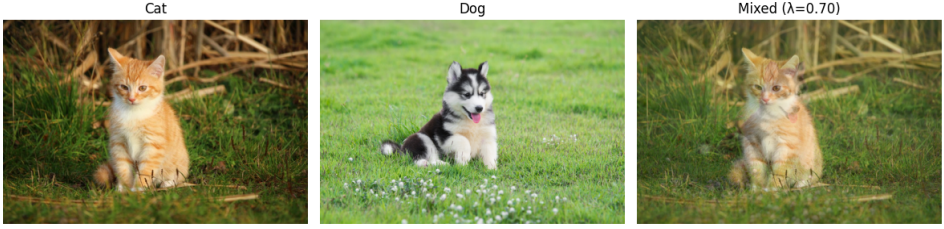
Data Augmentation

- **Definition**: Data augmentation involves generating new training samples by applying various transformations to existing data, helping improve model generalization.
- **Motivation**: Increases the diversity of the training dataset, which can reduce overfitting and improve the model's performance on unseen data.
- **Flips, rotations, and scaling:**

- **Random cropping**:

- **Color Adjustments:** Changes to colors or brightness

Mixup



- **Mixup** creates new training samples by blending pairs of images and their corresponding **labels**:

$$\tilde{\boldsymbol{x}} = (1-\lambda)\boldsymbol{x}_1 + \lambda \boldsymbol{x}_2,$$
$$\tilde{\boldsymbol{y}} = (1-\lambda)\boldsymbol{y}_1 + \lambda \boldsymbol{y}_2,$$

  where $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ are original training samples, and λ is the mixing factor drawn from a Beta distribution.
- This technique helps the model generalize by learning smoother transitions between classes.

Zhang, H., et al. *"Mixup: Beyond Empirical Risk Minimization."* ICLR 2018.

Outline

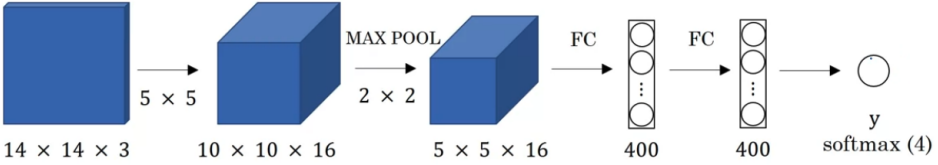
Sliding Window Detection

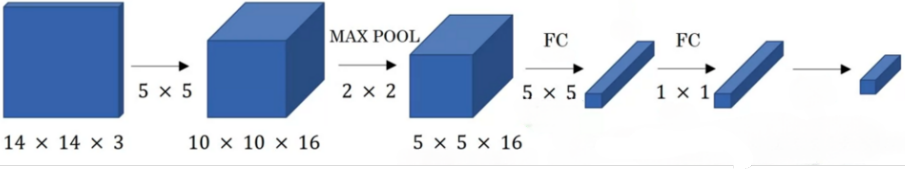
- Use a pre-trained classifier to identify cars in images.
- Scan the entire image with a sliding window, classifying each cropped section.

- **Problem**: This approach is computationally intensive due to the number of windows.

Turning Fully Connected Layers into Convolutional Layers

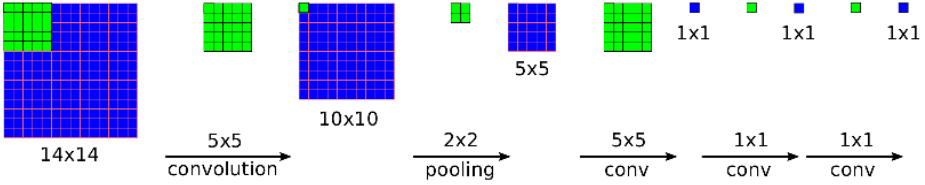
Fully connected layers can be implemented by convolutional layers:



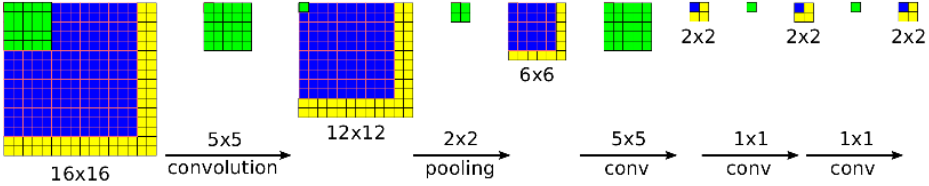


Convolution Implementation of Sliding Windows

ConvNets:



Detection:

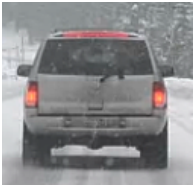


- During training, a ConvNet produces only a **single** spatial output
- When applied at test time over a *larger* image, it produces a spatial output **map**.
- However, with a fixed sliding window size, the bounding boxes may lack precision.

Sermanet, P., et al. "Overfeat: Integrated recognition, localization and detection using convolutional networks." ICLR 2014

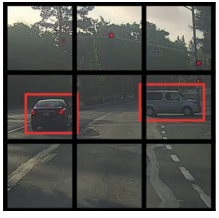
YOLO Algorithm

Object Localization



- **Input**: image with a single object
- **Output**: class probability and bounding box $\boldsymbol{y} = (p_c, b_x, b_y, b_w, b_h)$
- $p_c \in [0, 1]$: class confidence; (b_x, b_y): object center; b_w, b_h: weight and height

YOLO Algorithm Steps: Divide image into $S \times S$ **grid cells**



- Each cell predicts **class probabilities** and **bounding boxes**
- Use **squared loss** for bounding box predictions
- **Objectness confidence**: based on **Intersection over Union (IoU)**

$$\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

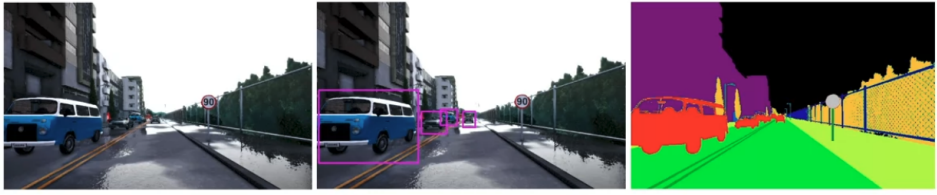
- In each cell, object center (b_x, b_y) uses local coordinates in $[0, 1] \times [0, 1]$
- b_w and b_h can be greater than 1 (relative to image size)
- Apply **non-max suppression** to retain highest confidence box

Redmon, J., et al. "You only look once: Unified, real-time object detection." CVPR 2016.

Outline

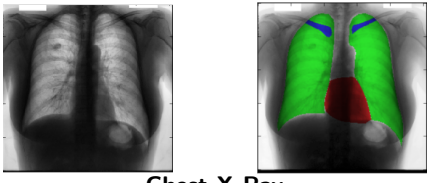
Semantic Segmentation

Detection v.s. Segmentation:

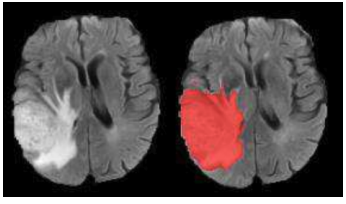


Input image | Object Detection | Semantic Segmentation

Successful Applications:



Chest X-Ray



Brain MRI

Novikov, et al. "Fully convolutional architectures for multiclass segmentation in chest radiographs." IEEE Tran Med Img 2018
Dong, et al. "Automatic brain tumor detection and segmentation using U-Net based fully convolutional networks.", MIUA2017

Semantic Labeling

Per-Pixel Class Labeling:

- Assign a class label to every pixel in the image.
- Output is an image of the same dimensions as the input.

Gao, et al. "SFSM: sensitive feature selection module for image semantic segmentation." Multimed. Tools Appl. 2023

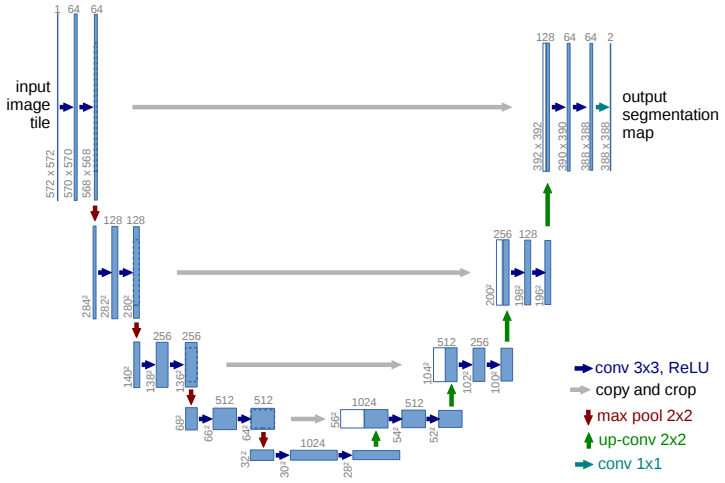
Transpose Convolution

Transpose Convolution: A transpose convolution (or a **deconvolution** or **up-sampling convolution**) is an operation that applies a filter to input data in a way that expands its spatial dimensions.

$$\underbrace{\begin{bmatrix} 3 & 0 \\ 1 & 5 \end{bmatrix}}_{\textbf{input } 2 \times 2} * \underbrace{\begin{bmatrix} 2 & 7 & 4 \\ 3 & 1 & 7 \\ 4 & 2 & 1 \end{bmatrix}}_{\textbf{filter } 3 \times 3} = \underbrace{\begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}}_{\textbf{feature map } 4 \times 4}$$

- The stride can be more than 1
- Padding is reversed by discarding boundary pixels.
- For overlaps, use averaging or summation.
- The filter represents patterns, with the input indicating where these patterns are detected.
- In unmax pooling, either duplicate pixels in the output or place the maximum value pixel while setting others to zero.

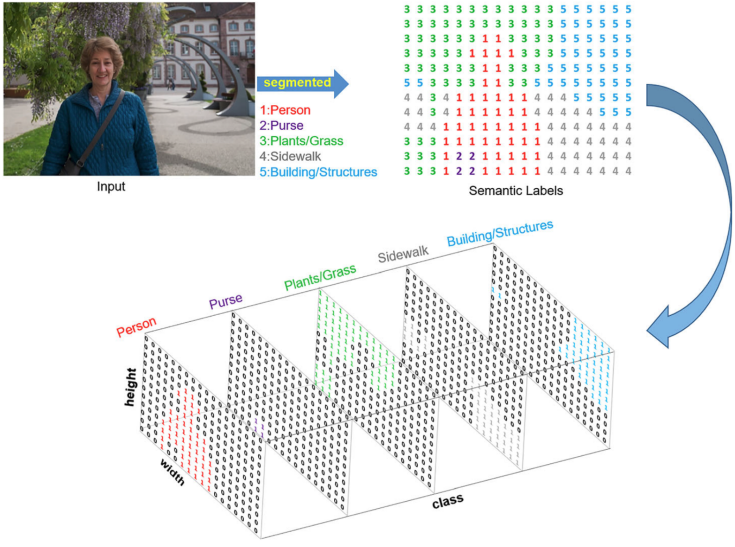
U-Net Architecture



- With skip connection, U-Net combines high-level abstract features (from deeper layers) spatial details (from earlier layers).

Ronneberger, et al. "U-net: Convolutional networks for biomedical image segmentation." MICCAI 2015

U-Net Output



Gao, et al. "SFSM: sensitive feature selection module for image semantic segmentation." Multimed. Tools Appl. 2023

Outline

Verification vs. Recognition

Verification

- **Input**: An image
- **Output**: Confirms if the image matches the claimed person

Recognition

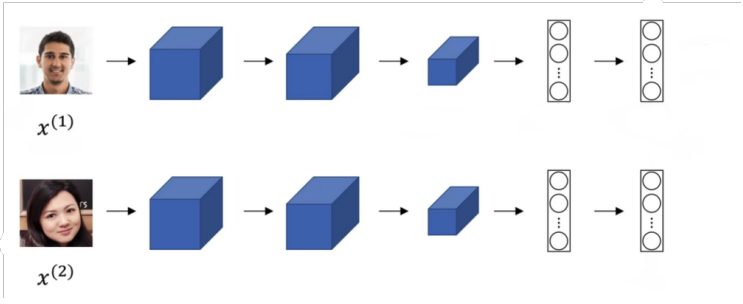
- **Database**: Contains K known identities
- **Input**: An image
- **Output**: Returns the person's ID if in the database; otherwise, not recognized

One-Shot Learning

- Learns from a single example to re-identify a person
- Challenges with new identities
- Solution: Use a similarity function to compare images

Siamese Network

Definition: A Siamese network is a neural architecture with two identical sub-networks that share weights and parameters.



- Here the ConvNet $\boldsymbol{f}(\boldsymbol{x})$ represents the **encoding** of input image $\boldsymbol{x}$
- Measure encoding divergence:

$$\hat{y} = \sum_{k=1}^{n} |f(\boldsymbol{x}_i)_k - f(\boldsymbol{x}_j)_k|$$

- Pass divergence through a logistic activation for binary classification: Same person?

Taigman, Yaniv, et al. "Deepface: Closing the gap to human-level performance in face verification." CVPR 2014.

Similarity Function and Triplet Loss

- **Similarity**: $d(\boldsymbol{x}, \boldsymbol{z}) := \|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{z})\|^2$ measures the difference between two images $\boldsymbol{x}$ and $\boldsymbol{z}$.
- Training uses image triplets: an **anchor** $\boldsymbol{a}$, a **positive** $\boldsymbol{p}$, and a **negative** $\boldsymbol{n}$.
- Objective:
$$d(\boldsymbol{a}, \boldsymbol{p}) + \alpha \leq d(\boldsymbol{a}, \boldsymbol{n})$$
where $\alpha > 0$ is the *margin*.
- Triplet loss:
$$\ell(\boldsymbol{a}, \boldsymbol{p}, \boldsymbol{n}) = [d(\boldsymbol{a}, \boldsymbol{p}) + \alpha - d(\boldsymbol{a}, \boldsymbol{n})]_+ ,$$
where $[x]_+ = \max\{x, 0\}$.
$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{(\boldsymbol{a}, \boldsymbol{p}, \boldsymbol{n})} \ell(\boldsymbol{a}, \boldsymbol{p}, \boldsymbol{n})$$
- Requires multiple images per person for learning.

Schroff, Florian, et al. "Facenet: A unified embedding for face recognition and clustering." CVPR 2015.

Outline

Neural Style Transfer

Content

Style

Transfered

- Randomly initialize an initial image
- Gradient descent to minimize the cost

$$\mathcal{L}(\boldsymbol{g}) = \alpha \mathcal{L}_{\text{content}}(\boldsymbol{g}, \boldsymbol{c}) + \beta \mathcal{L}_{\text{style}}(\boldsymbol{g}, \boldsymbol{s})$$

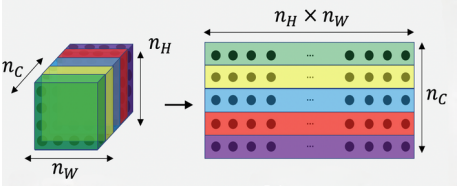
where

$$\mathcal{L}_{\text{content}}(\boldsymbol{g}, \boldsymbol{c}) = \sum_{\ell} \|\boldsymbol{a}^{\ell}(\boldsymbol{g}) - \boldsymbol{a}^{\ell}(\boldsymbol{c})\|^2$$

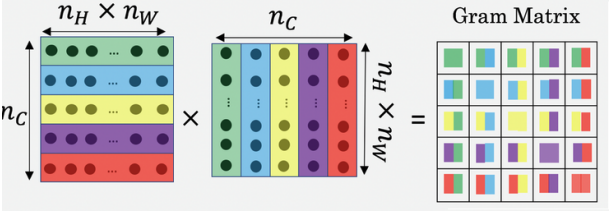
Gatys, Leon A., et al. "Image style transfer using convolutional neural networks." CVPR 2016.

Style Matrix

- The **style** is defined as how correlated the activations are across different channels.



- The style matrix $\boldsymbol{G}^{\ell} \in \mathbb{R}^{n_c \times n_c}$ is defined by: $\boldsymbol{G}^{\ell}_{k\bar{k}} := \left\langle \boldsymbol{a}^{k}, \boldsymbol{a}^{\bar{k}} \right\rangle$, $\forall k, \bar{k} \in [n_c]$.



- Then the style cost is

$$\mathcal{L}_{\text{style}}(\boldsymbol{g}, \boldsymbol{s}) = \sum_{\ell} \frac{1}{n \times n \times n_c} \|\boldsymbol{G}^{\ell}(\boldsymbol{g}) - \boldsymbol{G}^{\ell}(\boldsymbol{s})\|^2$$