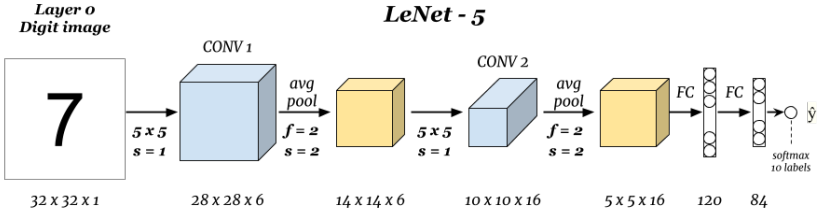
Learning with CNNs, Part II

Tianxiang (Adam) Gao

School of Computing
DePaul University

Outline

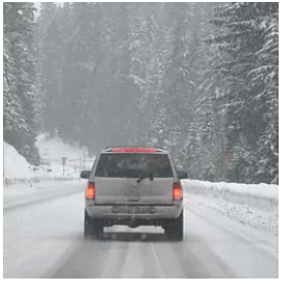
Recap: Convolutional Neural Networks (CNNs)

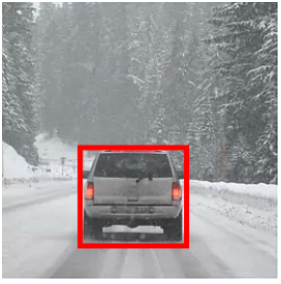


- **Convolution and hierarchical representation:** It slides small **filters** over inputs to extract patterns; stacked layers progressively combine simple patterns into higher-level features.
- 1×1 **Convolution**: Learns high-level patterns by combining multiple basic patterns.
- **Transfer Learning**: Fine-tune a large, pretrained model on a smaller dataset using a lower learning rate to learn task-specific features.
- **Data Augmentation**: Increases dataset diversity and reduces overfitting (e.g., flips, random cropping, color adjustments, mixups).
- **Semantic Segmentation**: Assigns a class label to each pixel. U-Net combines lower- and higher-level features using an encoder-decoder architecture.
- **Face Recognition**: Learns a similarity function explicitly (via triplet loss) or implicitly (via siamese networks).

Outline

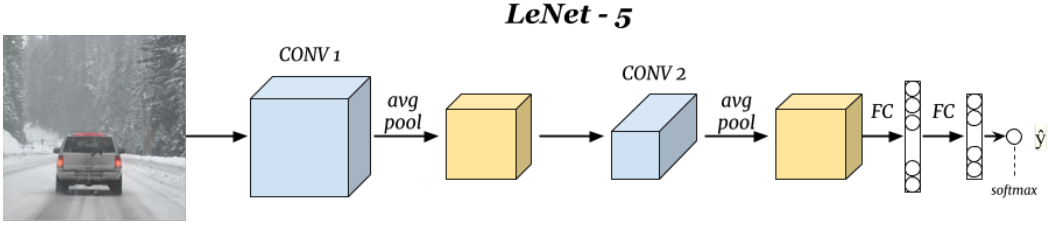
Object Localization





- **Input**: An image containing a **single object**
- **Output**: Class probability and bounding box

Classification with Object Localization

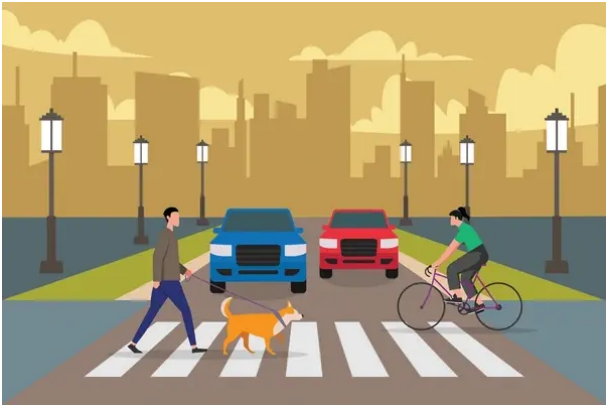


- **Bounding box**:
  - $p_c \in [0, 1]$: confidence score indicates whether an object presences;
  - (b_x, b_y): object center;
  - b_w, b_h: width and height
- Use **squared loss** for bounding box predictions
- **Multi-classification**: the output becomes $\boldsymbol{y} = (p_c, b_x, b_y, b_w, b_h, c_1, c_2, \cdots, c_N)$

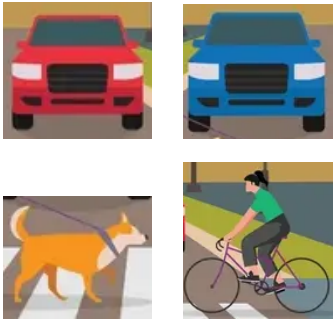
Sliding Window Detection

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

Original Image



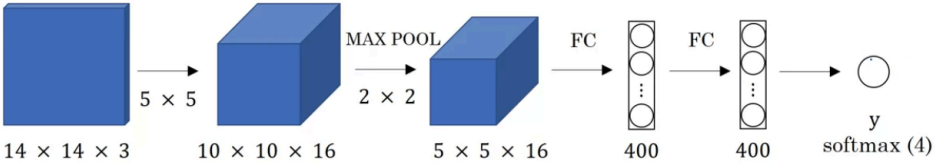
Cropped Objects



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

Convolutional Layers as Fully Connected Layers

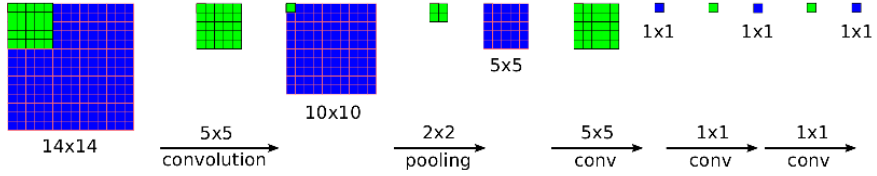
Fully connected layers can be implemented by convolutional layers:



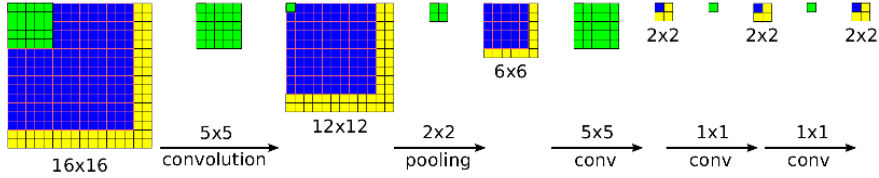
Note: Flattening a feature map and feeding it into an FC layer is equivalent to applying a convolution, since both perform linear combinations over all inputs with an equal number of parameters.

Convolution Implementation of Sliding Windows

ConvNets:



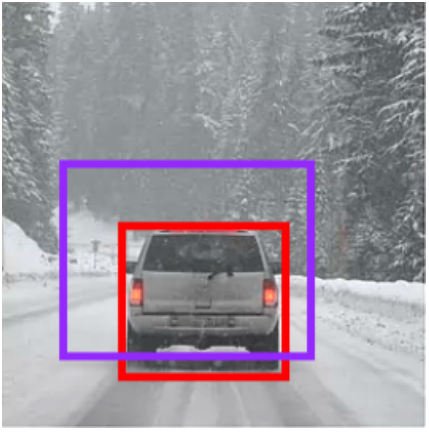
Detection:



- **Each cell** in the **final feature map** encodes a **high-level representation** of a **local receptive field** in the original input image, which the detection head uses to predict object presence and location.
- With an added **detection head**, the CNN predicts **objectness, class, and bounding box** for each region, achieving sliding-window detection in one convolutional pass.

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

Evaluating Object Localization using IoU

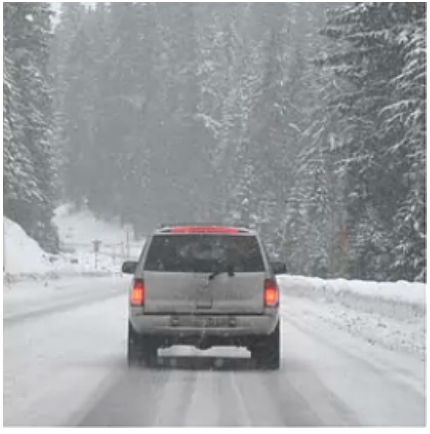


- **Objectness confidence**: based on **Intersection over Union (IoU)**

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

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

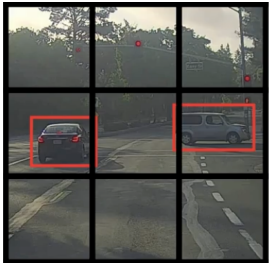
Non-Maximum Suppression (NMS) Algorithm



- Each cell predicts and outputs $y = (p_c, b_x, b_y, b_w, b_h, c_1, \cdots, c_N)$.
- **Filter** boxes by confidence: keep $p_c \geq 0.6$.
- **Sort** remaining boxes by p_c in descending order.
- Pick **highest** p_c box and output that as a prediction
- **Suppress** any box with $\mathrm{IoU} \geq 0.5$ to it; repeat.

YOLO Algorithm: IoU and Non-Maximum Suppression (NMS)

Problem: Multiple bounding boxes may be predicted for the same object.



- Choose the bounding box with the highest confidence score p_c as the best detection.
- Remove all other highly overlapping boxes with **Intersection over Union (IoU)** > 0.5 compared to the best box.

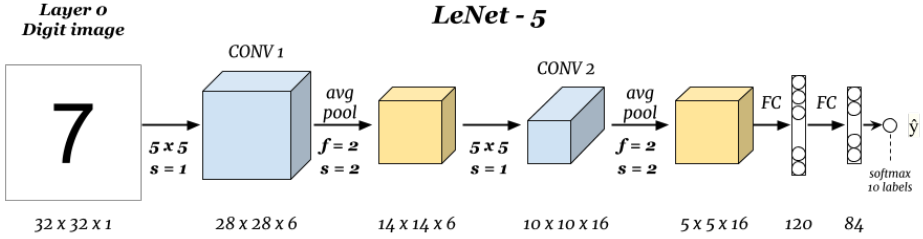
$$\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- Keep low-overlap boxes as they likely correspond to different objects (e.g., other cars).

Redmon, J., et al. "You Only Look Once: Unified, Real-Time Object Detection." CVPR 2016.

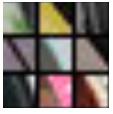
Outline

Understanding CNNs Through Visualization



How to Visualize CNN Filters:

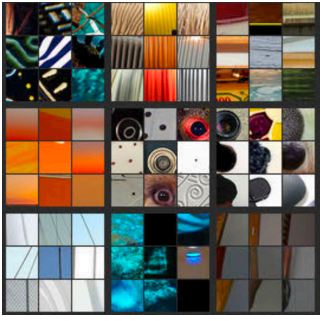
- Process many input images through a **well-trained** CNN.
- Select a **specific convolutional filter** in a layer.
- Find **9 different images** where this filter had the **highest activations** in the feature maps.
- Use **DeconvNet** to **reconstruct** the input regions responsible for these activations.
- This produces **9 reconstructed patches**, revealing what pattern this filter detects.



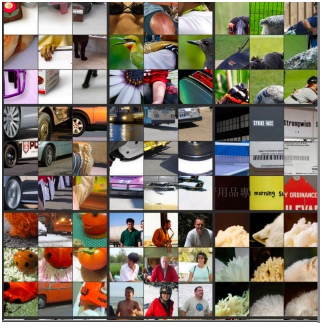
Zeiler, Matthew D., and Rob Fergus. "Visualizing and Understanding Convolutional Networks." ECCV 2014.

Visualizing Deep Layers

Layer 1



Layer 2



Layer 3

Neural Style Transfer

Content

Style

Transfered

- Randomly initialize an initial image $\boldsymbol{g}$
- Optimize $\boldsymbol{g}$ using gradient descent to minimize the total loss:

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

where

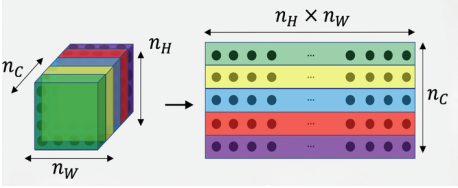
$$\mathcal{L}_{\text{content}}(\boldsymbol{g}, \boldsymbol{c}) = \sum_{\ell} \|\boldsymbol{a}^{\ell}(\boldsymbol{g}) - \boldsymbol{a}^{\ell}(\boldsymbol{c})\|^2$$

Here, $\boldsymbol{a}^{\ell}$ represents the activations of layer ℓ in a pre-trained CNN, *e.g.*, VGG.

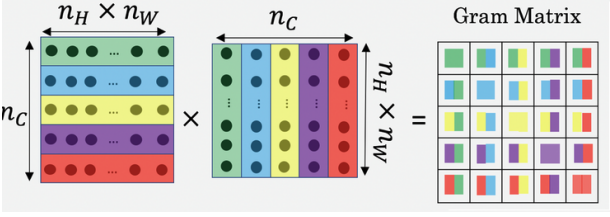
Gatys, Leon A., et al. "Image style transfer using convolutional neural networks." CVPR 2016.

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

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