**DETECTION ALGORITHM**

**\*Image Classification and Localization:**

ConvNets categorize images into specific classes like cars, pedestrians, or backgrounds. This foundational task leads to classification with localization, where ConvNets not only identify objects but also predict bounding box coordinates for precise

localization within images. Supervised learning methods and target label formulations play a crucial role in training ConvNets for classification with localization.

**\*Landmark Detection:**

landmark detection is the specialized task within object detection. ConvNets excel in identifying specific points of interest on objects or faces, providing coordinates for

landmarks. This capability is instrumental in applications such as face recognition,

emotion detection, and fine-grained spatial analysis.

**\*Sliding Windows Detection Algorithm:**

The traditional sliding windows detection algorithm is introduced as a systematic

approach to object detection. ConvNets process image regions with varying window sizes, scanning the entire image for objects. However, this method has computational challenges, particularly with ConvNets, where processing each window independently can be resource-intensive.

**\*Convolutional Implementation for Efficiency:**

Convolutional implementation emerges as a solution for efficient object detection.In this approach significantly reduces computation while maintaining high detection accuracy. The transition from traditional sliding windows to convolutional implementations highlights benefits such as scalability and computational efficiency.

**\*Implementing Convolutional Algorithm for Object Detection**

Transitioning from fully connected layers to convolutional layers in neural networks for object detection.

An ex from Ng’s course is given below involves processing 14x14x3 images through 5x5 filters, leading to a 10x10x16 output, and then max pooling to 5x5x16.

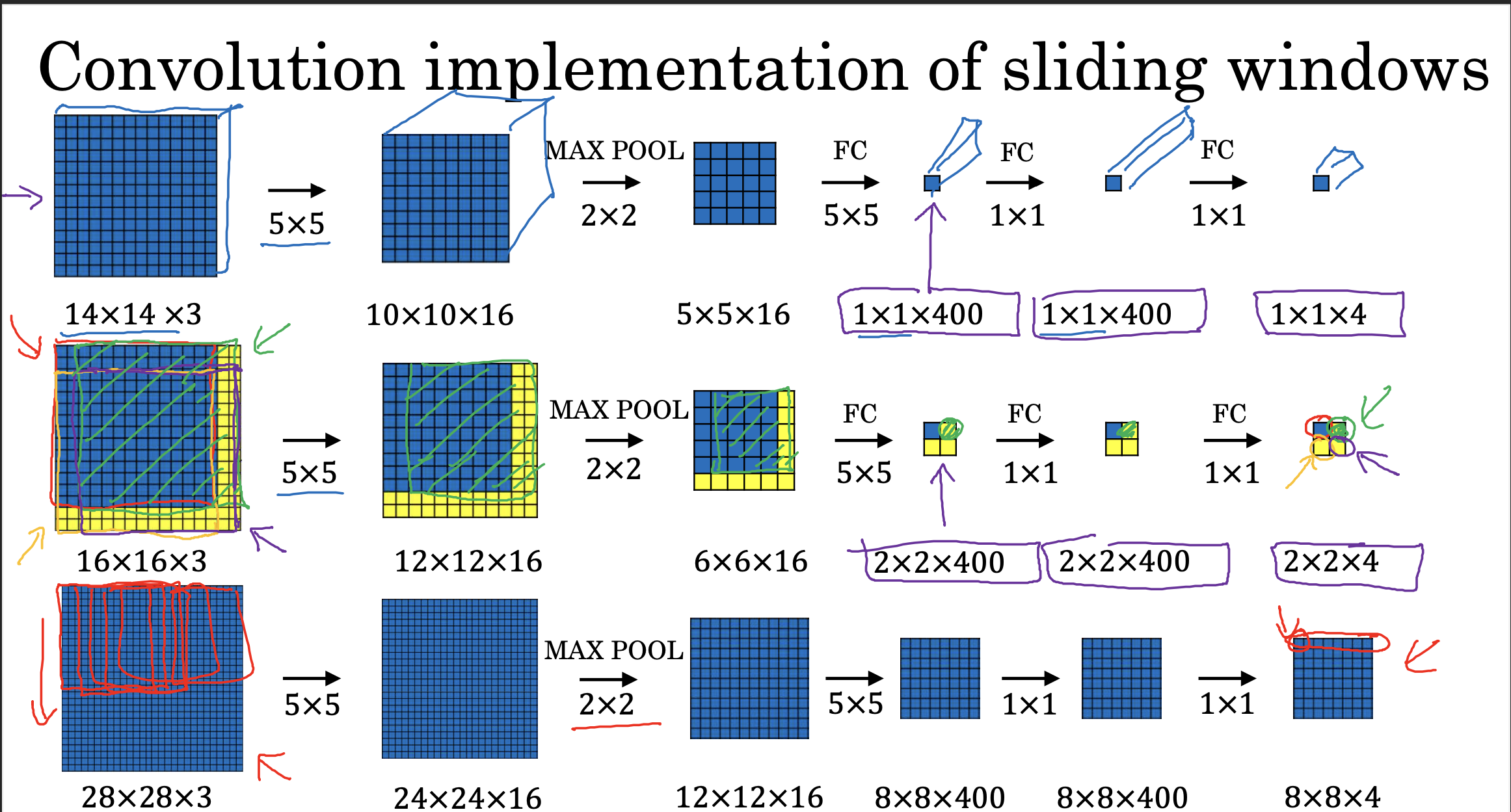
The transition to convolutional layers treats the fully connected layer as a 5x5 filter,

creating a 1x1x400 volume.

This process demonstrates a convolutional implementation for the entire image,

optimizing object detection algorithms.

In the diagram below the entire network is explained with dimensions.



**\*Convolutional Implementation of Sliding Windows Object**

**Detection**

The convolutional implementation of sliding windows object

detection using a 14x14x3 image input is done. It contrasts traditional sliding windows algorithms with a more efficient convolutional approach. By sharing computations across multiple regions of the image, convolutional implementation significantly reduces redundant computations, making object detection more efficient.

**\*Bounding Box and YOLO Algorithm**

YOLO (You Only Look Once) algorithm is used for object

detection, a bounding box serves as a important component for

localizing objects within an image. When YOLO processes an

image using a grid system (such as 3x3 or 19x19), each grid cell can predict bounding boxes along with class probabilities.

Here's how a bounding box fits into the YOLO algorithm:

Bounding Box Prediction: For each grid cell that contains an

object, YOLO predicts bounding boxes (bx, by, BH, BW) relative to that grid cell. These bounding box predictions specify the

position (bx, by) and size (BH, BW) of the object within the cell.

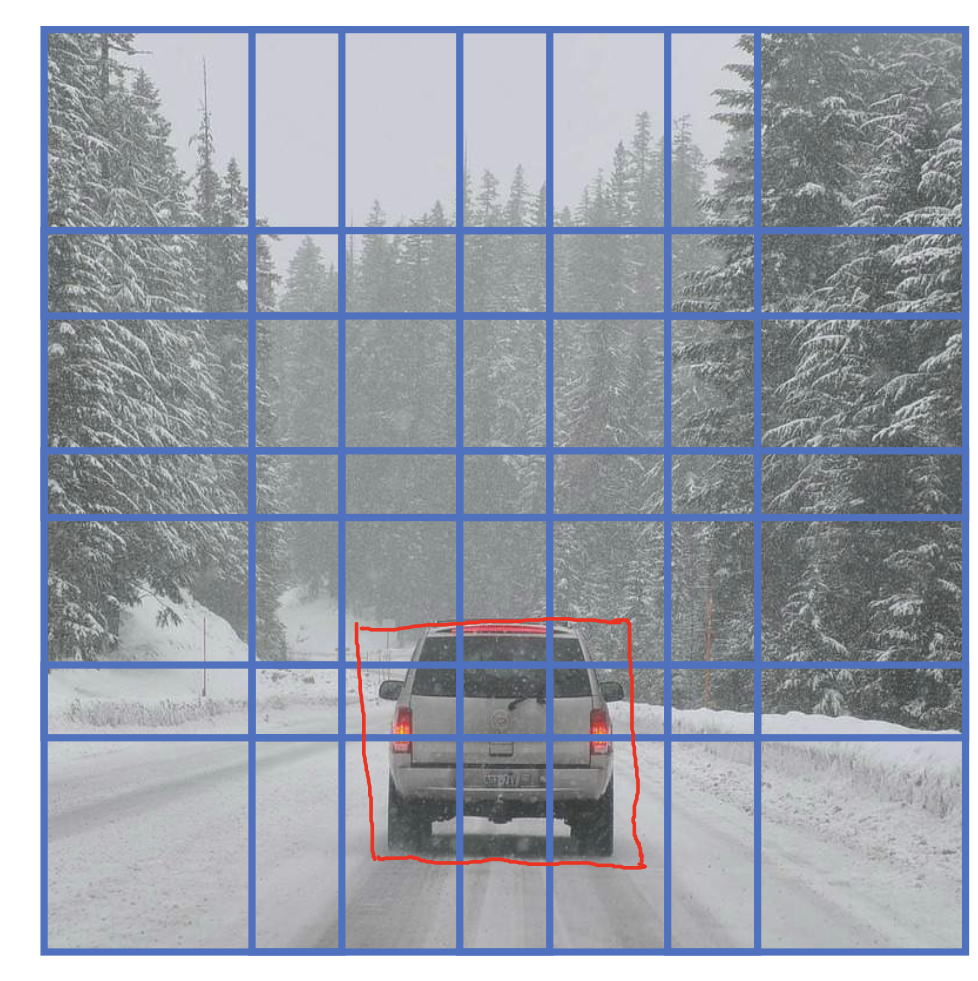
Object Localization: The bounding boxes predicted by YOLO

localize where objects are located within the image. The (bx, by) coordinates determine the center of the box relative to the grid cell, while BH and BW specify the height and width of the box.

Bounding Box Accuracy: The accuracy of bounding box

predictions is evaluated using metrics like Intersection over Union (IoU). IoU measures how well the predicted bounding box overlaps with the ground truth bounding box, providing a quantitative measure of localization accuracy.

Visual Representation(given below): Bounding boxes are visually represented as rectangles overlaid on the image, highlighting the region where YOLO has detected objects. These bounding boxes play a crucial role in object localization and subsequent classification by the algorithm.



\***Non-Max Suppression (NMS):**

Mainly it cleans up multiple detection by looking at the

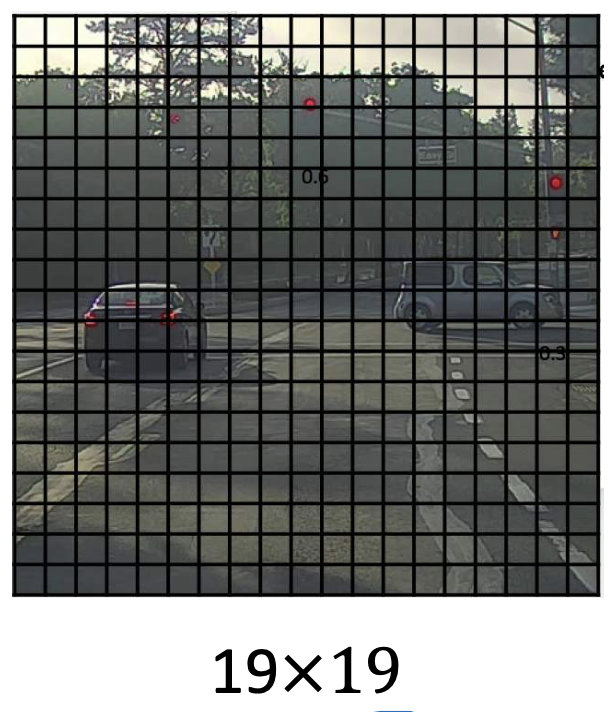
probabilities of various detection.

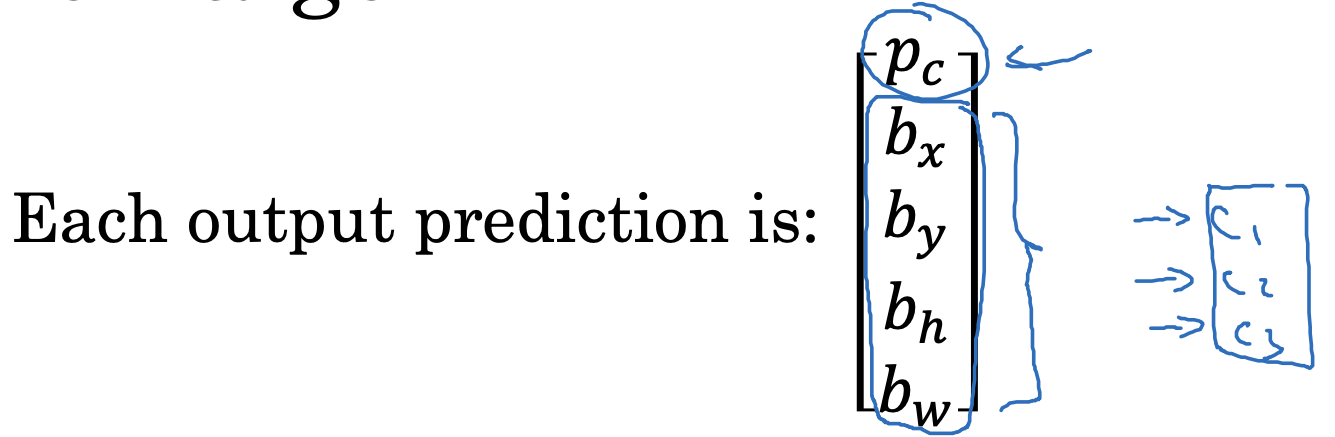
Firstly it selects the largest detection value (confident detection)

And all other detection with high IoU gets suppressed.

Below is an example of 19x19 grid with output prediction as

follows:



****

Discard all boxes with pc<=0.6(lets say)

Now for rest boxes,

Select the box with highest pc  this will be the output prediction

Now discard rest with IoU>=5 with box output in previous step.

\***Anchor Box:**

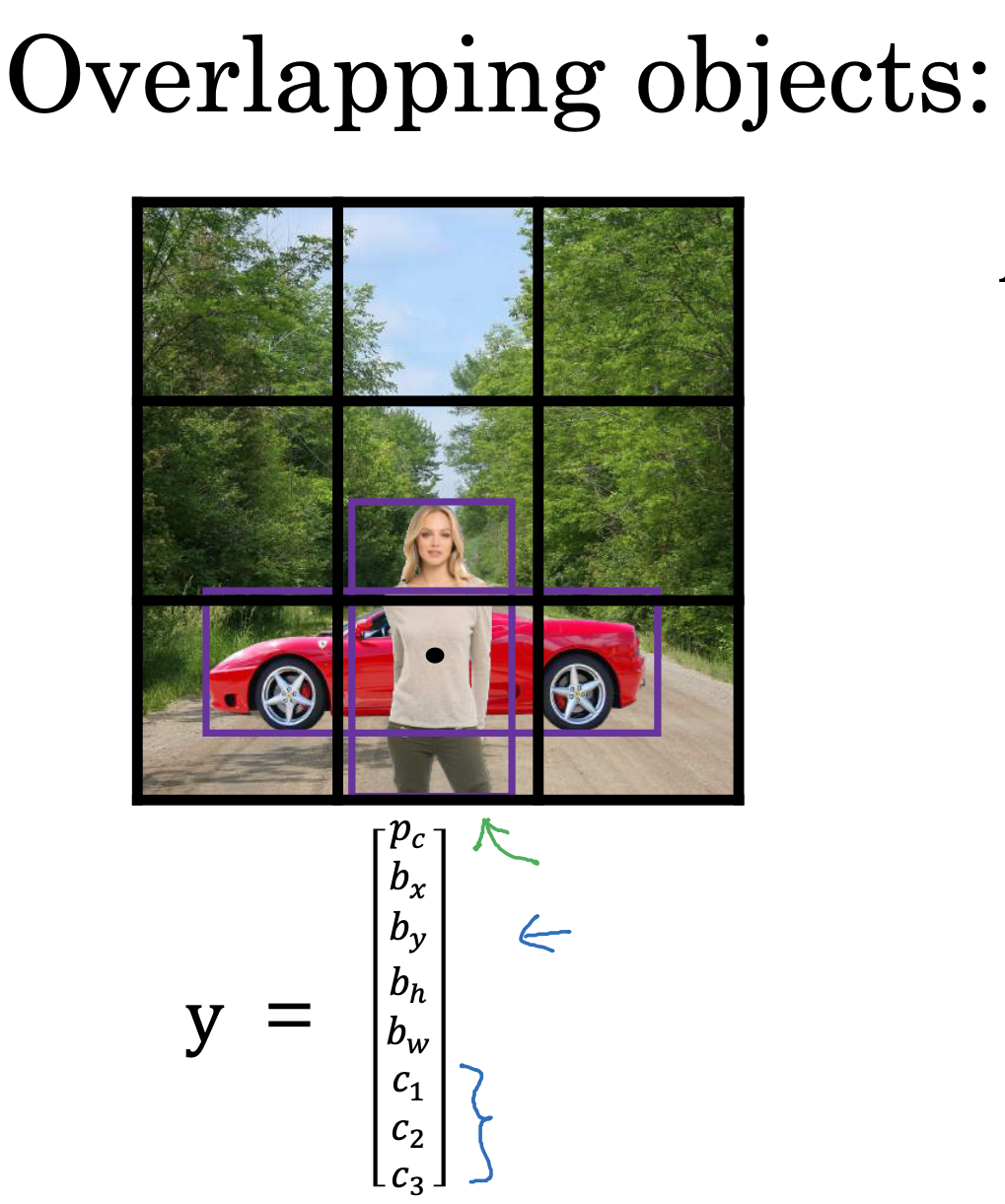
They are used to set different predictions present in the same input .we can define different anchor boxes.

For the eg below

We use 2 anchor boxes

The y cross label is changed and vector component are repeated twice for both anchors.

Now we can encode the respective anchor box for the 2 sets of component of y accordingly.

Previously each object in training image is assigned to grid cell that contains the midpoint of object(output y:3x3x8) but now with 2 anchor box each object in training image is assigned to grid cell that contains the midpoint of object and anchor box for grid cell with highest IoU.(output y:3x3x16 ,3x3x2x8)

**\*Intersection Over Union (IoU)**

IoU is a metric used to evaluate the accuracy of bounding box predictions. It measures the overlap between predicted and ground truth bounding boxes. Higher IoU values indicate more accurate predictions.

Here's a breakdown of how IoU works:

Definition: IoU is calculated as the ratio of the area of intersection between the predicted bounding box and the ground-truth bounding box to the area of their union.

Calculation:

Intersection Area (IA) = Area where predicted box and ground-truth box overlap

Union Area (UA) = Total area covered by both predicted and ground-truth boxes

IoU = IA / UA

Threshold: By convention, IoU values greater than 0.5 are often considered acceptable. This means that if the IoU is greater than or equal to 0.5, the predicted bounding box is considered correct.

Evaluation: Higher IoU values indicate better accuracy in localization. An IoU of 1 means perfect overlap, where the predicted and ground-truth bounding boxes are identical.

IoU is commonly used inalgorithms like Non-Max Suppression (NMS) to refine object detection results.

**\*Semantic segmentation with U-net**

**\*Semantic segmentation**

Semantic segmentation is a computer vision technique that

involves classifying each pixel in an image into a predefined category. Unlike object detection, which identifies and localizes objects using bounding boxes, semantic segmentation provides detailed and precise classification at the pixel level.

**Application:**

Self-Driving Cars:

Object Detection: Identifies objects like other cars, pedestrians, and obstacles using bounding boxes.

Semantic Segmentation: Labels every pixel to determine drivable surfaces, lane markings, and other crucial elements. This helps in precise navigation and safety measures by providing detailed environmental understanding.

Medical Imaging:

Chest X-Rays: Segmentation of lungs, heart, and clavicle bones to help diagnose conditions by highlighting specific anatomical regions.

Brain MRI: Automatic segmentation of brain tumors to assist radiologists and surgeons in diagnosis and surgical planning

**\*U-Net Architecture for Semantic Segmentation**

The U-Net is anarchitecture for semantic segmentation due to its ability to capture both context and fine details through its unique structure.

**U-Net Structure—**

Input: image with dimensions h x w x channels

Contracting Path (Encoder):

Repeated convolutions followed by ReLU activation and max-pooling for downsampling.

Reduces dimensions while increasing depth (number of channels), capturing high-level features.

Bottleneck:

Deepest part of the network with smallest dimensions but highest feature richness.

Expanding Path (Decoder):

Uses transpose convolutions (deconvolutions) to upsample feature maps.

Upsampling steps followed by concatenation with corresponding feature maps from the contracting path via skip connections.

Further convolutions refine upsampled features.

Skip Connections:

Directly connect output of each layer in contracting path to its

corresponding layer in the expanding path.

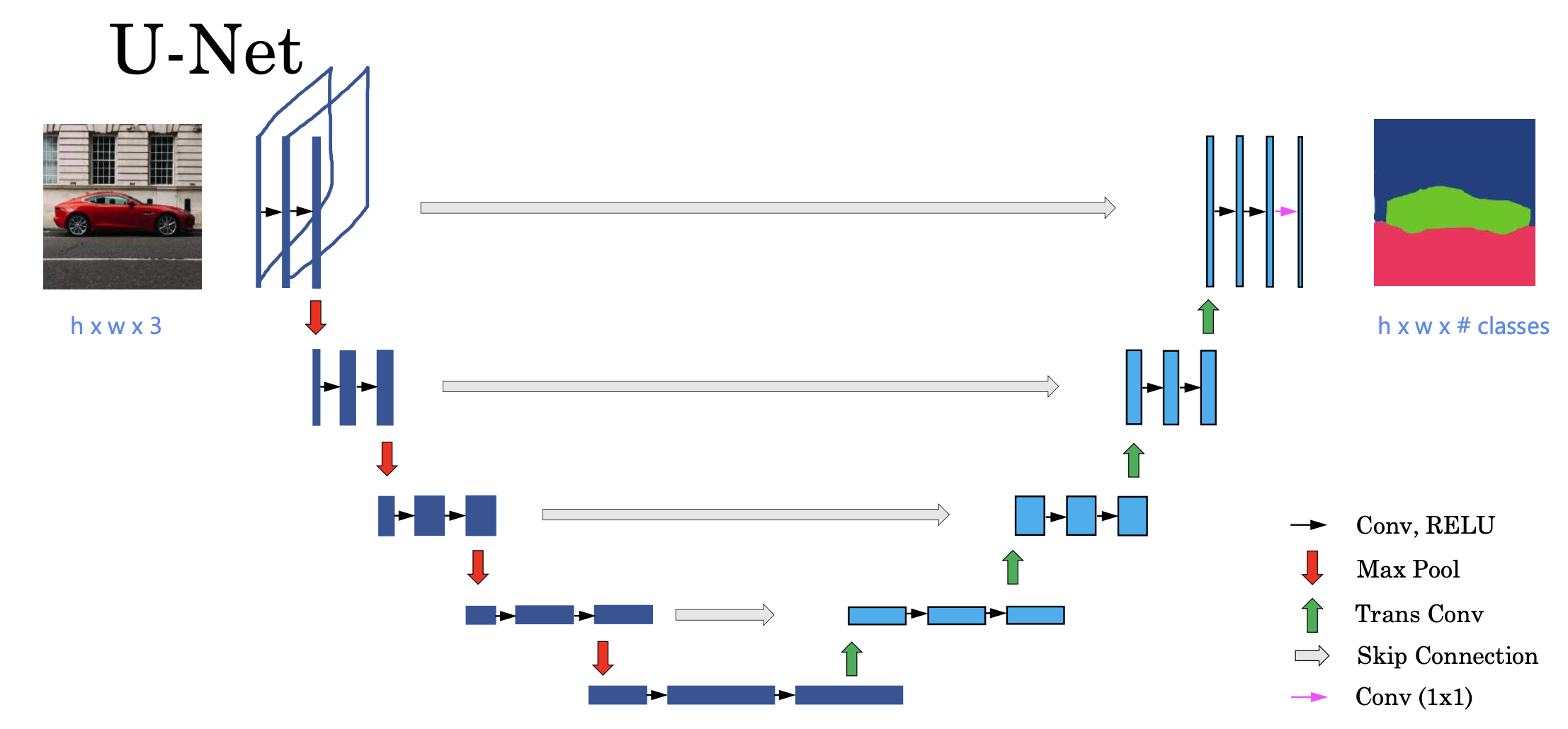
Preserve high-resolution information from early layers, combining with high-level features during upsampling.

Final Output Layer:

1x1 convolution to map features to desired number of classes for each pixel.

Output: Segmentation map with dimensions:

H x w x classes.

U-net dimensional diagram is given below-

**\*Transpose Convolution**

Transpose convolution, also known as deconvolution, is an

operation used to upsample feature maps in a neural network. It effectively increases the spatial dimensions of the input feature map.

Steps-

Input and Filter:

Take a small input feature map and a filter. For ex, consider a 2x2 input convolved with a 3x3 filter.

Now,

Place the filter on the output space rather than the input space.

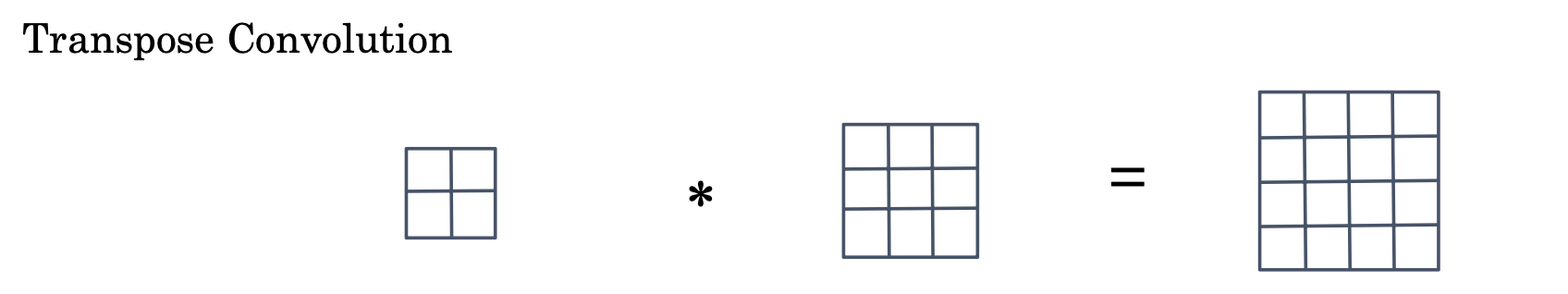
Each pixel value from the input is multiplied by the entire filter, and the result is added to the corresponding positions in the output feature map.

Overlapping values in the output feature map are summed to

ensure that spatial details are retained.

Example Calculation:

Given a 2x2 input and a 3x3 filter, the transpose convolution process can increase the output size to 4x4. This process helps in enlarging the dimensions while ensuring that the output retains necessary details.



**FACE RECOGNITION & NEURAL STYLE TRANSFER**

**\*Face verification vs. face recognition**

**Verification**

• Input image, name/ID

• Output whether the input image is that of the

claimed person

**Recognition**

• Has a database of K persons

• Get an input image

* Output ID if the image is any of the K persons (or “not recognized”)

**\*One-Shot Learning**

One-shot learning in face recognition aims to identify individuals using only a single example image per person, addressing the challenge of limited training data.

Explaining it with an example -

Lets consider there are 4 employee A,B,C,D

A new person E comes and system must identify them based on single image per person

Challenges with Traditional Methods:

ConvNet with a softmax layer struggles with small datasets.

Adding new employees requires retraining, which is impractical.

Solution to above is learning a similarity function

**\*Similarity Function (d)**

Train a neural network to compare two images and output a similarity score.

Small score for images of the same person, large score for different

people.

**\*Face Verification Process:**

Compare the new image with each database image using d

Use a threshold τ to determine if images match:

Score < τ: Same person.

Score > τ: Different people.

**\*Siamese network**

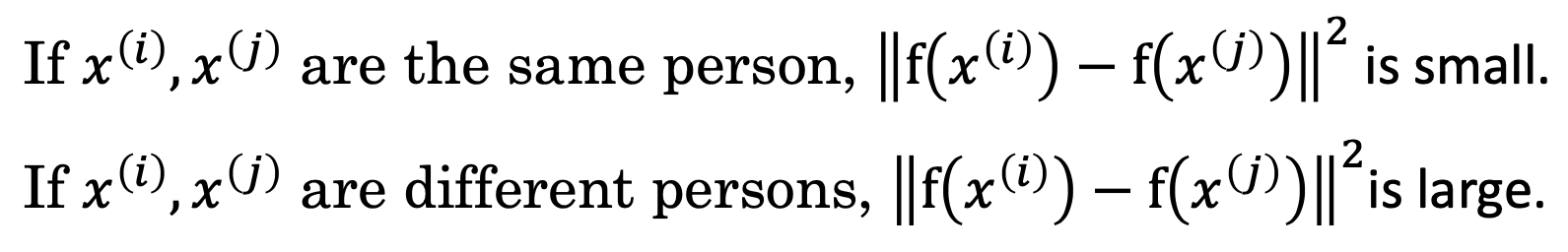
To measure the similarity or difference between two faces using a function d

Siamese Network: Uses two identical convolutional neural networks (CNNs) with shared parameters to process two input images.

Feature Vector: Each network generates a 128-dimensional feature vector (encoding) for the input image.

Input image x1 & x2 into the same network

Output the vector f(x1) and f(x2)

Now compute d(x1,x2) = distance between f(x1)&f(x2)

Here x(I) is x1 and x(j) is x2.

**\*Triplet loss**

To effectively learn the parameters of a neural network for face recognition, we use a method called the triplet loss function. This approach involves comparing pairs of images to ensure the network can differentiate between faces of the same person and faces of different persons.

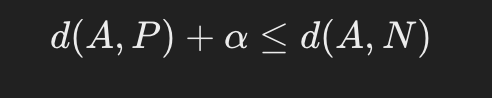
The triplet loss function relies on three images:

Anchor (A): The reference image.

Positive (P): An image of the same person as the anchor.

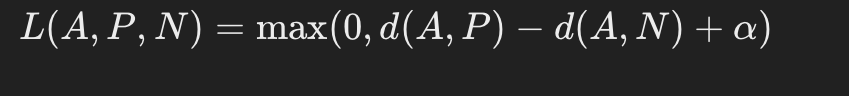
Negative (N): An image of a different person.

The objective is to train the network so that the distance between the anchor and the positive image (d(A, P)) is smaller than the distance between the anchor and the negative image (d(A, N)) by a margin α.

We want —

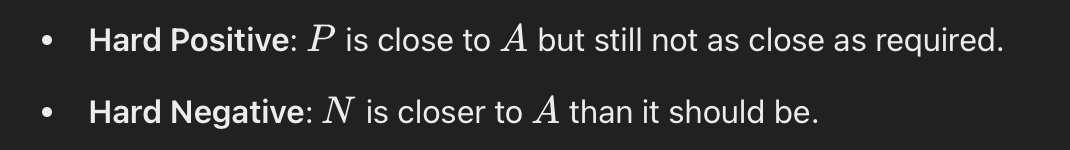
————(1)

(|| f(A) - f(P) ||^2)/d(A,P) + α <= (|| f(A) - f(N) ||^2)/d(A,N)

The loss function can be written as-

The loss is 0 if condition (1) is met otherwise its positive .

During training, A,P,N are selected randomly but we should choose triplets that are hard to train on.



**\*Neural Style Transfer**

Neural Style Transfer is a technique that uses a Convolutional Neural Network to generate a new image by combining the content of one image with the style of another. By optimizing the generated image to minimize the content difference from the content image and the style difference from the style image.

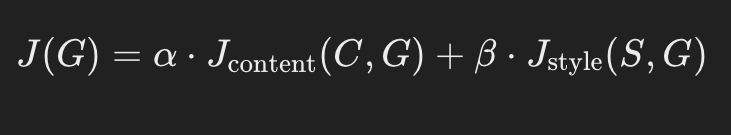
**\*Cost function**

We will take—

Content image (c)

Style image (s)

&get Generated image (G)

Define cost function J(G) for how good the image is-

α,β=hyperparameters

To find the generated image G-

Initiate G randomly

G-100 x 100 x 3

Now use gradient descent to minimise J

**\*Content cost function**

Use hidden layer l to computer content cost

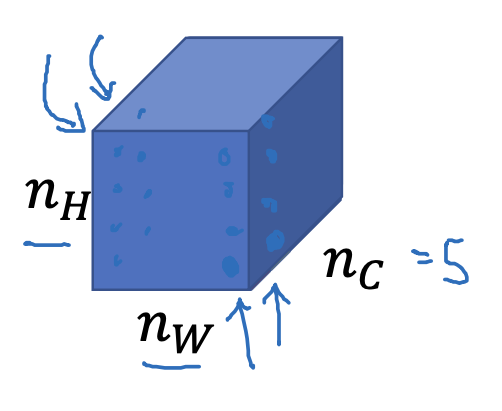
We will use any pre trained Conv net(VGG )

Let a[l]C and a[l]G activation of image in layer l, if they are same both the image have same content

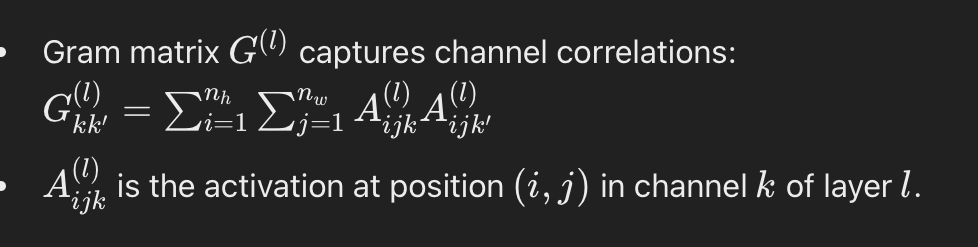
**\*Style cost function**

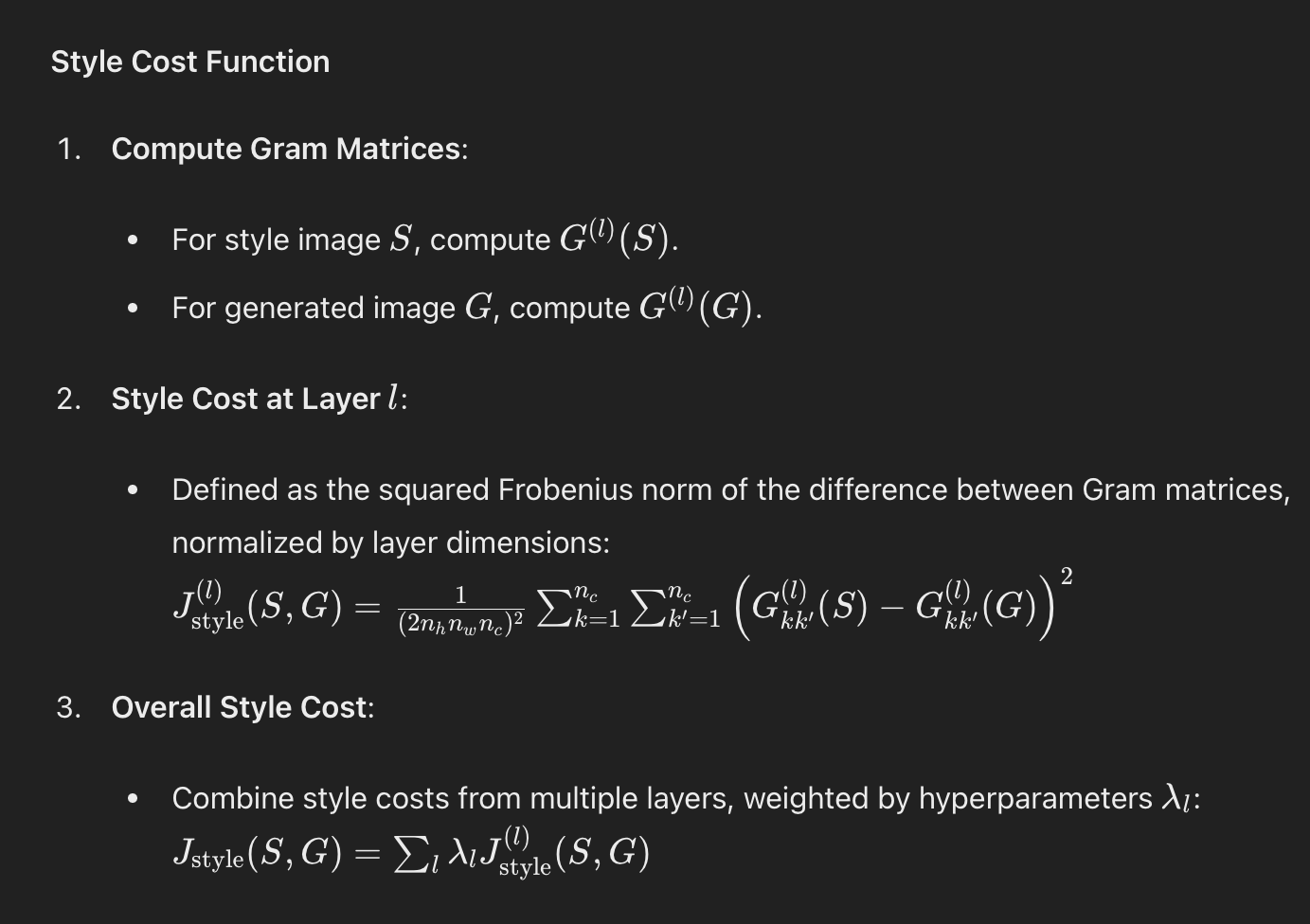
we are using layer l’s activation to measure “style.”

Define style as correlation between activations across channels.

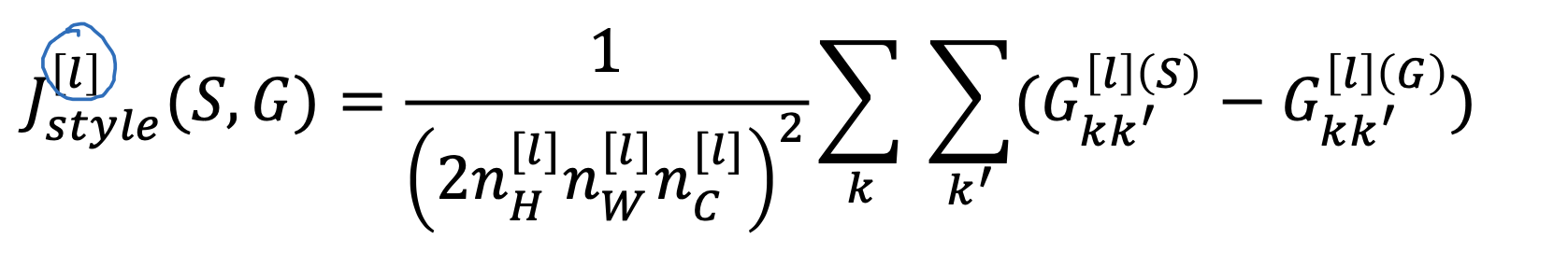
Activations form a 3D block as given below

Style is captured by correlation b/w feature channels

Style matrix -



Overall style cost function—>



**\*1D Convolutions**

Input: 14

Filter: 5

Output: 10(after convolution)

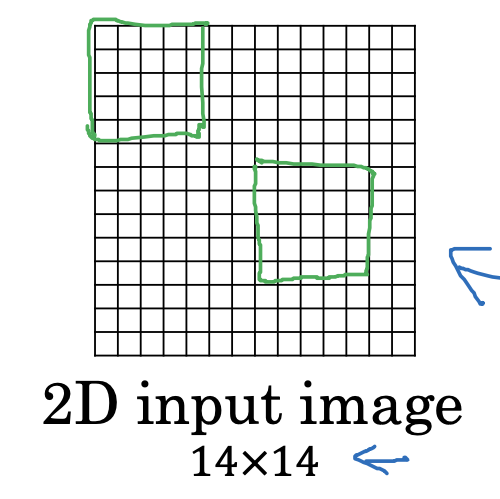
Multiple Channels and Filters:

For a single-channel input (e.g., one EKG lead), and 16 filters, the output becomes

10×16

Subsequent layers might convolve

10×16 input with another 5-length filter across 16 channels, resulting in

6×32 if using 32 filters.

**\*3D Convolutions**

3D Data Ex: CT scans provide 3D volumes of the body.

Input: 14×14×14 (height, width, depth)

Filter: 5×5×5

Output:

10×10×10 (after convolution)

Multiple Channels and Filters:

For a single-channel input (e.g., CT scan), and 16 filters, the output becomes

10×10×10×16.

Subsequent layers might convolve

10×10×10×16 input with another

5×5×5 filter across 16 channels, resulting in

6×6×6×32 if using 32 filters.

**REVIEW QUESTIONS**

(Not able to find anything relevant related to global padding)

**1.Where is Global Padding used?**

In web development it is applied in CSS to maintain spacing around HTML elements.

And used in UI design.

**2.translation equivariance & invariance**

The translation equivariance is obtained by means of the convolutional layers. In fact, if the input image is translated to the right by a certain amount, the feature maps generated by convolutional layers are shifted by the same amount and direction.

CNNs designed for image classification are translation invariant since if we translate the input, then the output label will not be influenced. Translation invariance is obtained in CNNs by means of the [pooling layers](https://www.baeldung.com/cs/ai-convolutional-neural-networks#2-pooling-layers). The pooling operation is usually applied to the feature map generated by preceding convolutional layers and non-linear activation functions. Pooling is the substitution of features in a neighborhood with representative statistics, the max or the mean generally. Hence, the location of the original feature is disregarded.

CNNs are not naturally equivariant and invariant to

rotation, scaling, and affine transformations. Hence, [data augmentation](https://www.baeldung.com/cs/ml-data-augmentation) techniques must be used to make CNNs more robust to such geometric transformations.

**3.Which hyperparameters are learnable in CNN?**

certain hyperparameters learnable:

**Learning Rate** Schedules: Adaptive methods like Adam, RMSprop, and learning rate schedules dynamically adjust the learning rate during training.

**Adaptive Kernel Sizes:** Some advanced models allow kernel sizes to be adaptive, letting the network learn optimal filter sizes during training.

**Techniques like Neural Architecture Search (NAS)**

**Dynamic Weight Agnostic Neural Networks (DWANN)**