Table of Contents

[Style Transfer and Object Detection 2](#_Toc92407961)

[Introduction to Style Transfer 2](#_Toc92407962)

[Style Loss and the Gram Matrix 4](#_Toc92407963)

[Loss Function 7](#_Toc92407964)

[Style Transfer Notebook 8](#_Toc92407965)

[Object Detection - I 10](#_Toc92407966)

[Object Detection - II 13](#_Toc92407967)

[Summary 14](#_Toc92407968)

# Style Transfer and Object Detection

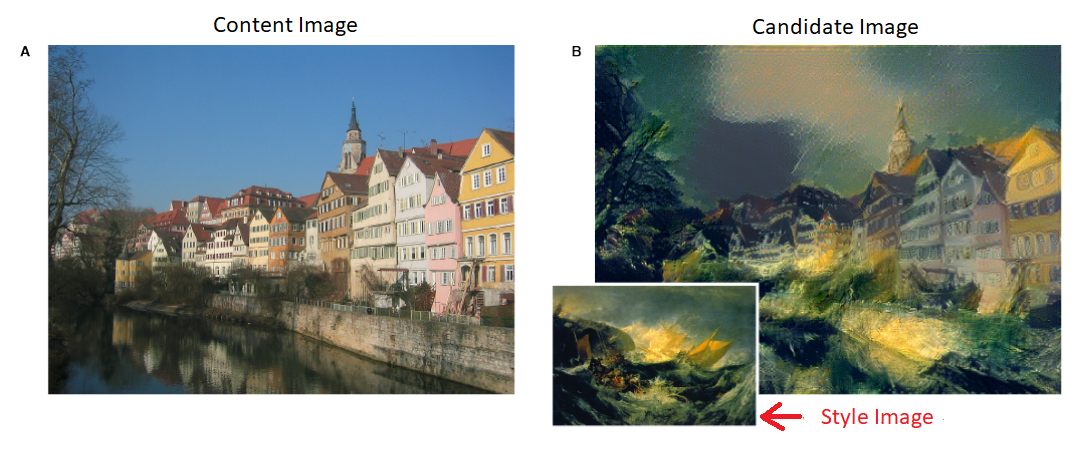
## Introduction to Style Transfer

In this session, we will discuss an 'artistic' application of Convolutional Neural Networks called **image style transfer**. Style transfer was first introduced in the paper “[A Neural Algorithm of Artistic Style](https://arxiv.org/pdf/1508.06576.pdf)” published in 2015 by LA Gatys et al.

 Style transfer is an application of **transfer learning** where you have a 'content image' and a 'style image' and the objective is to transfer the style from the style image to the content image**.**

Let's look at the basic idea of style transfer in this lecture.

**To summarise, the objective is to transfer the 'style' from the style image and render that style into the content image. An example of content and style images is shown below (taken from the original paper) - the 'style' from the style image is transferred to the content image to produce the 'candidate image'.**

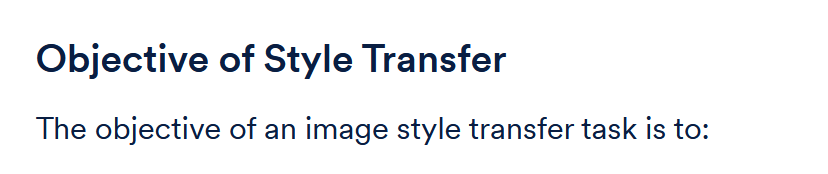


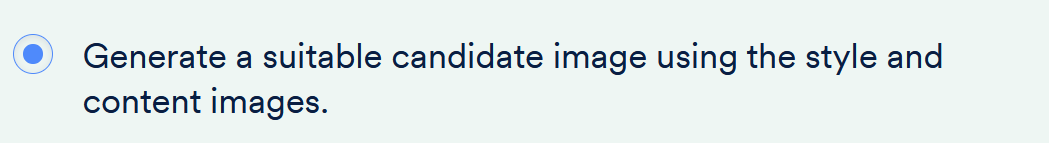
Candidate Image that combine the content of a photograph(content image) with the style(style image)

The notations used to denote the images are as follows:

1. Content Image: T
2. Style Image: S
3. Candidate Image: C

Note that there are many possible candidate images one can generate using the same content and style images, and the task is to **generate a 'suitable' candidate image**. That is, the candidate image should resemble the content from the content image and the style from the style image. In the next few lectures, you will study how this is done by the underlying algorithm.

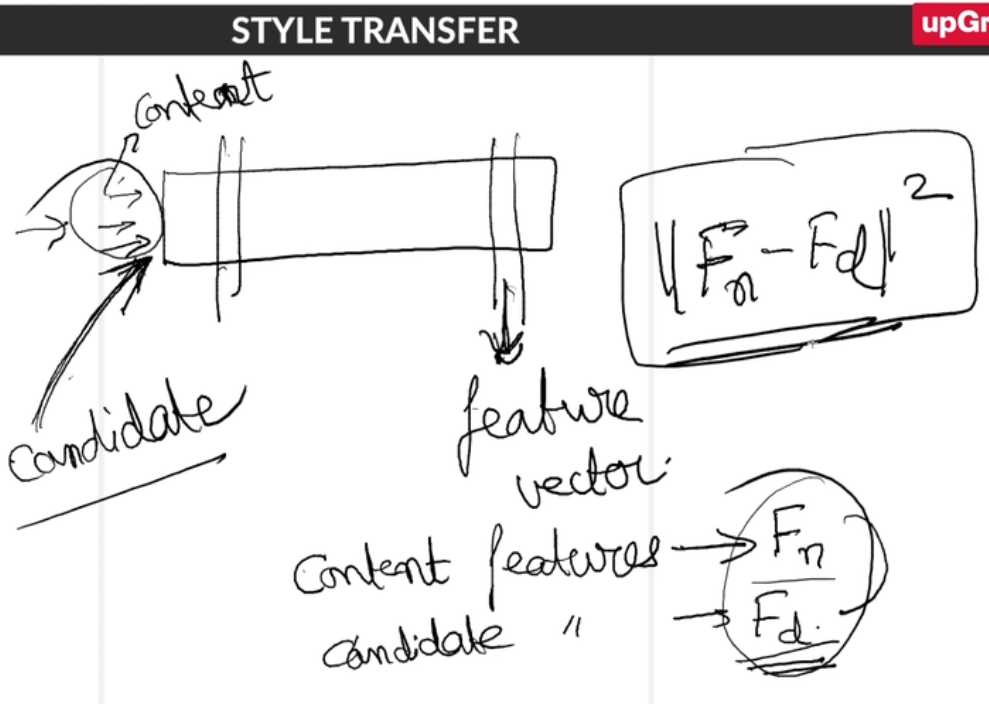




Thus, we want to generate a suitable **candidate image** which resembles the 'content' of the **content image** and has a 'style' similar to the **style image**.

But what does 'style' even mean? How do we extract the style of an image and write an algorithm which can transfer the style to the content?

If you think about it, what we call 'style' is actually a lot of things - the colour combinations, the textures, the intensities of various colours, etc. Let's see how we can translate the notion of content and style into a mathematical form.



To summarise, the 'content' part of the loss function ensures that the feature vector corresponding to the content and candidate images is the same.

When you feed the content image to a pre-trained network such as VGGNet, you'll get a **content feature vector** which is a representation of the content image in the last layers of the network. Likewise, if you feed a candidate image, you'll get a **candidate feature vector** which is a representation of the candidate image.

It is important to note that **the training task is to learn the candidate image**. That is, during training, the individual pixel values of the candidate image are learnt. You randomly initialise the candidate image and then update the pixels using backpropagation.

**Notation:**

Let’s denote the content feature vector by **Fn** and candidate feature vector by **Fd** (as produced by a deep network such as the VGGNet). For **Fn**to be close to **Fd**, the **L2** norm of **Fn** and **Fd** must be as small as possible. Therefore, we define **content loss** as:

**Content Loss = ||Fn−Fd||2**

Now, this is only the content loss. You also want that the candidate image absorbs the style from the style image. In the next segment, you will study how the second component of the loss function - the style loss.

## Style Loss and the Gram Matrix

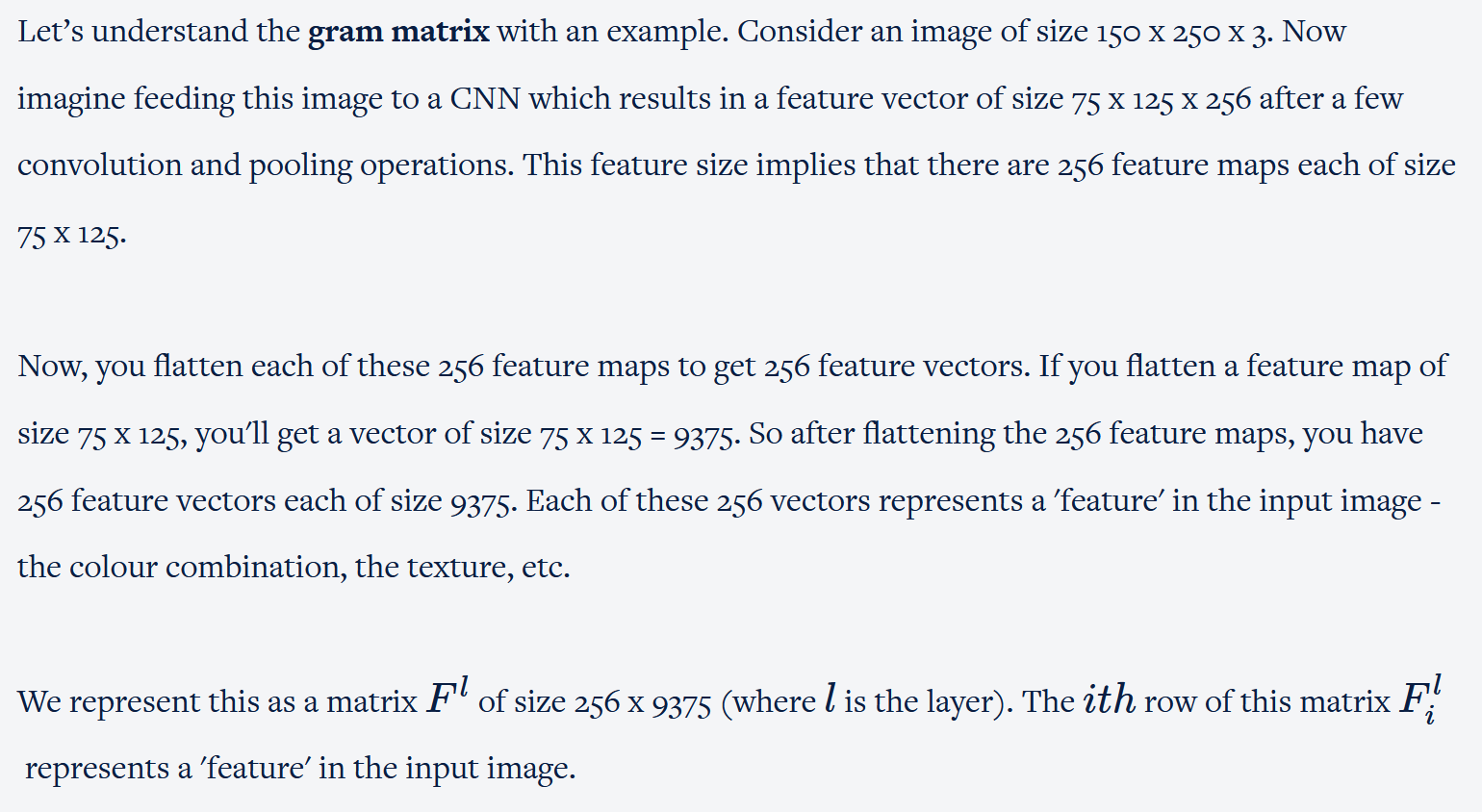
In the previous segment, you studied how content loss is computed. The second component of the loss function is the **style loss**. Let's understand the intuition behind style loss.

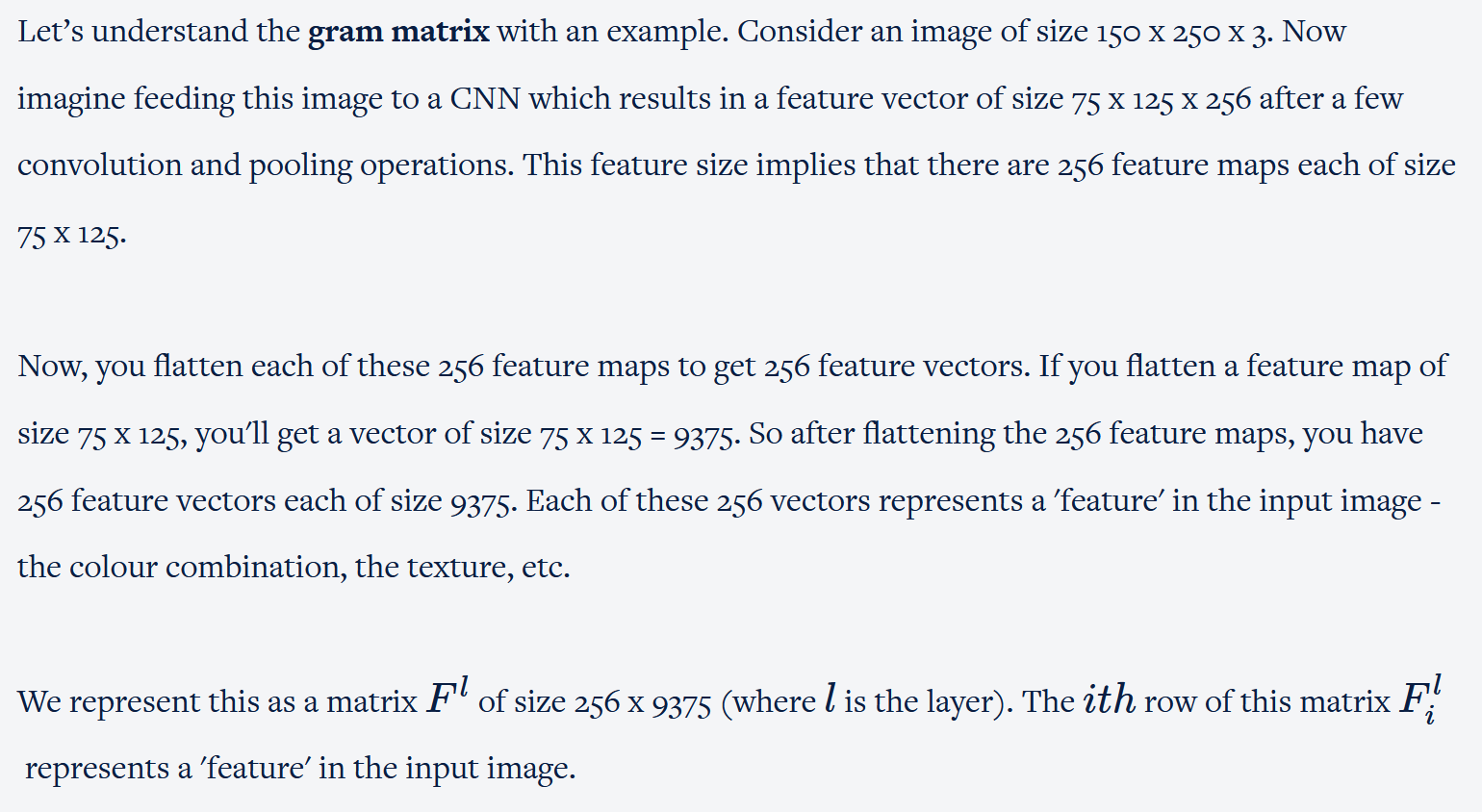
Using the example of a web browser, we saw that:

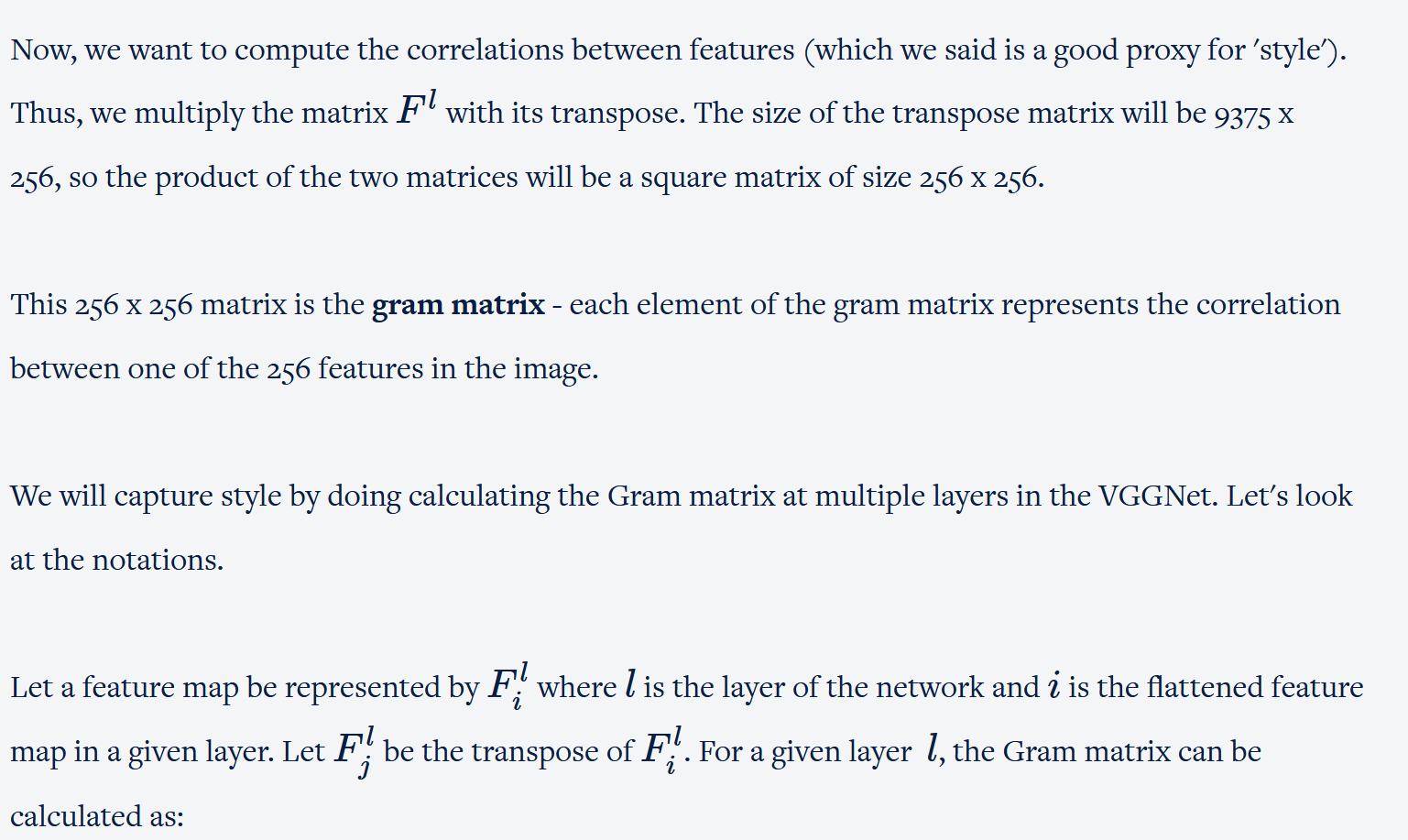
* **Whether a button is present on the screen is governed by the content.**
* **How the button appears relative to the menu bar is governed by the style: the theme of the browser.**

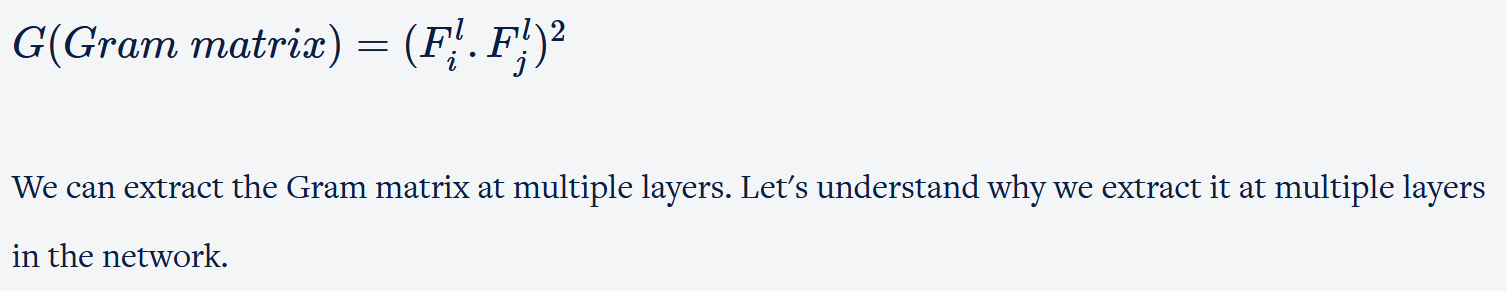
In general, style can be thought of as 'how different features of an image interact with each other'. For example, if you increase the red tint in the upper part of the image, how this change affects the blue tint in the lower part of the image is something that's governed by the style. Similarly, how the texture of one part on an image changes relative to the colours in another part could be governed by style.

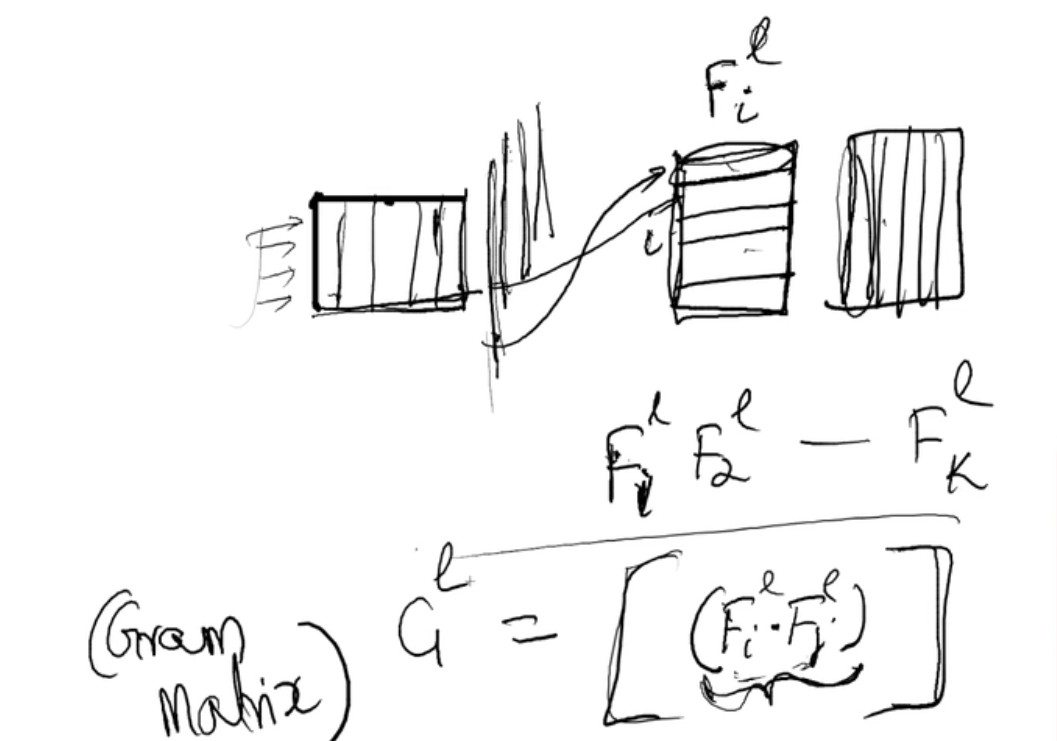
One can see style as the correlation between the different features of an image. Now, take a while to brainstorm about how you can represent style mathematically and incorporate into a loss function.

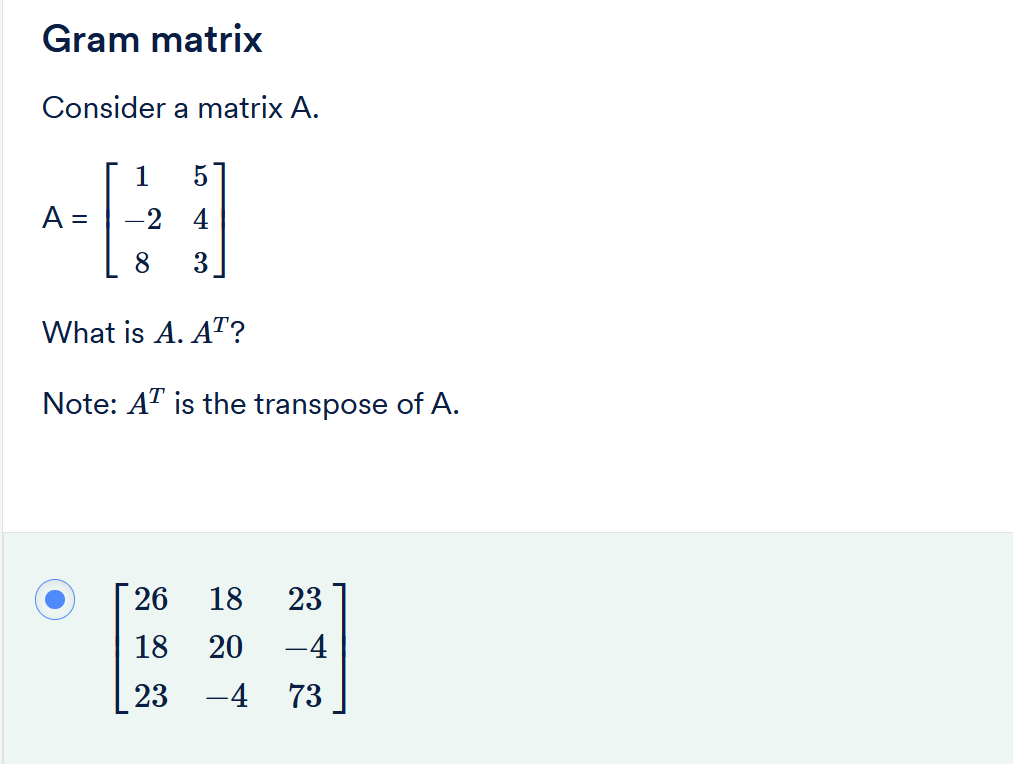


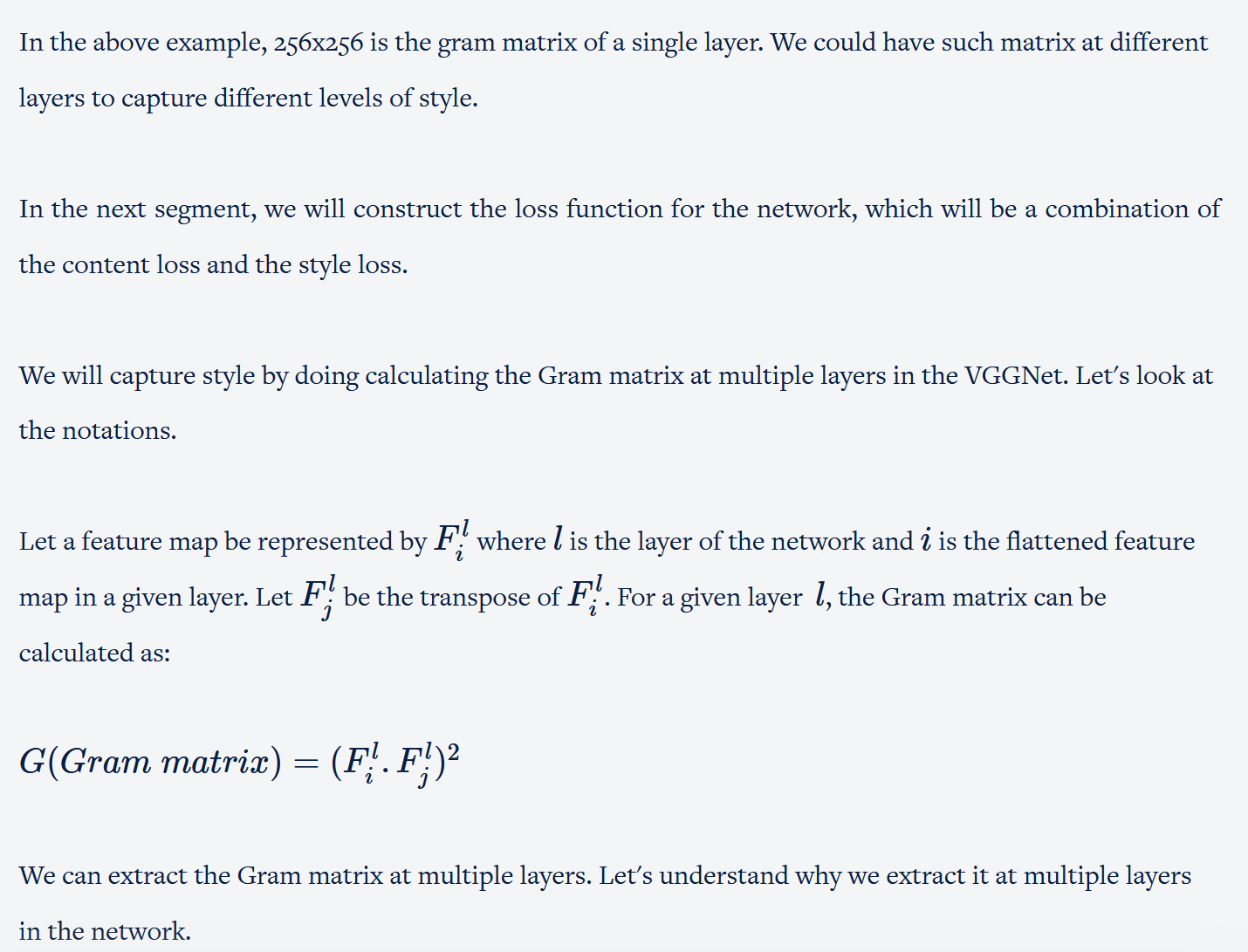










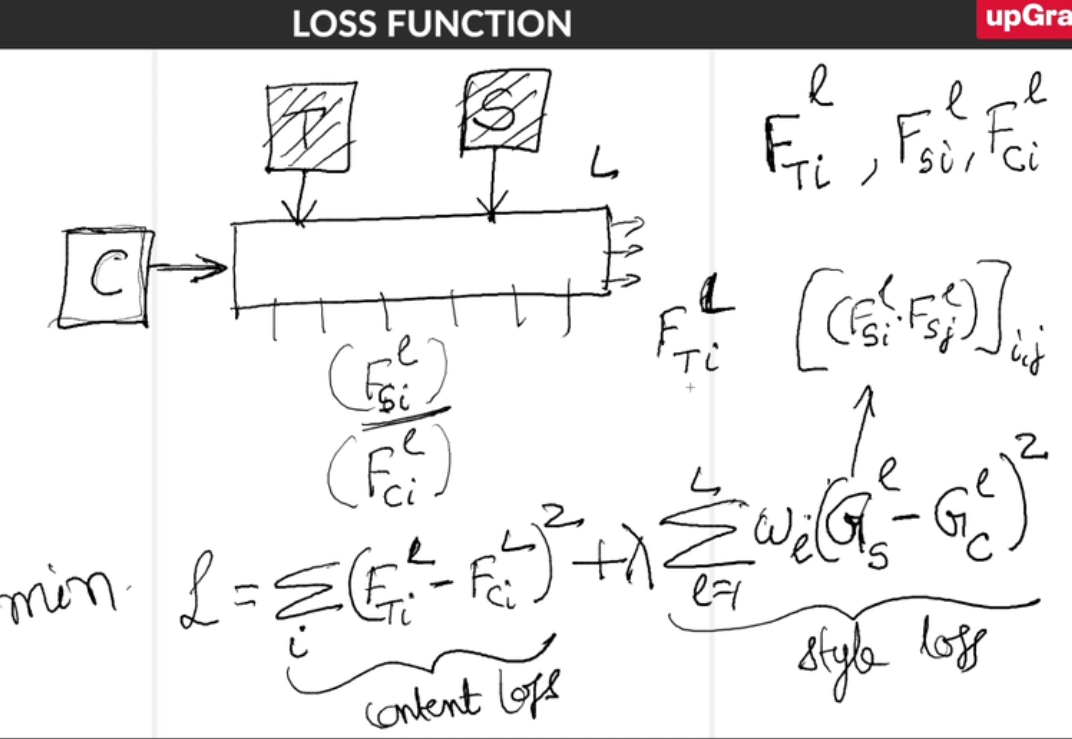


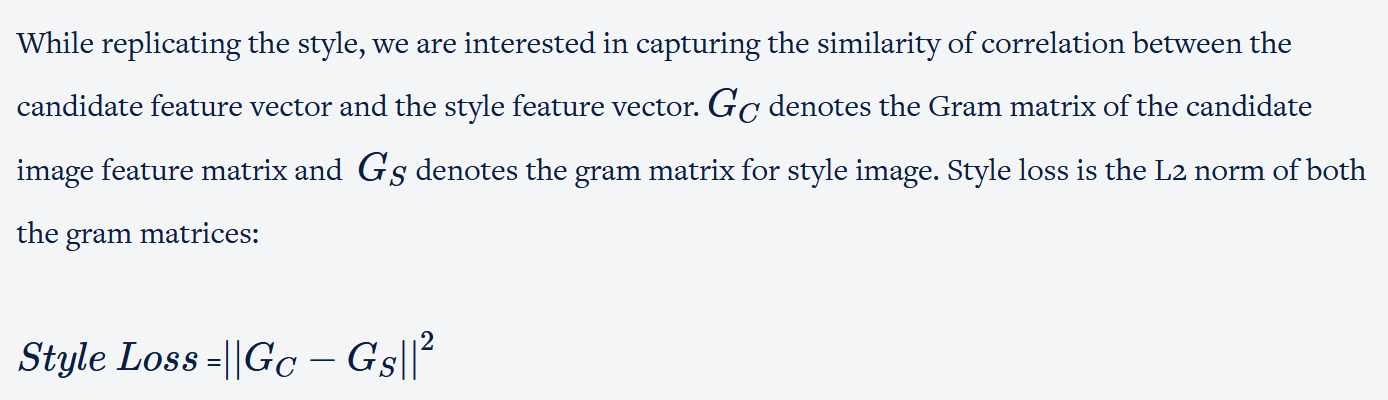
## Loss Function

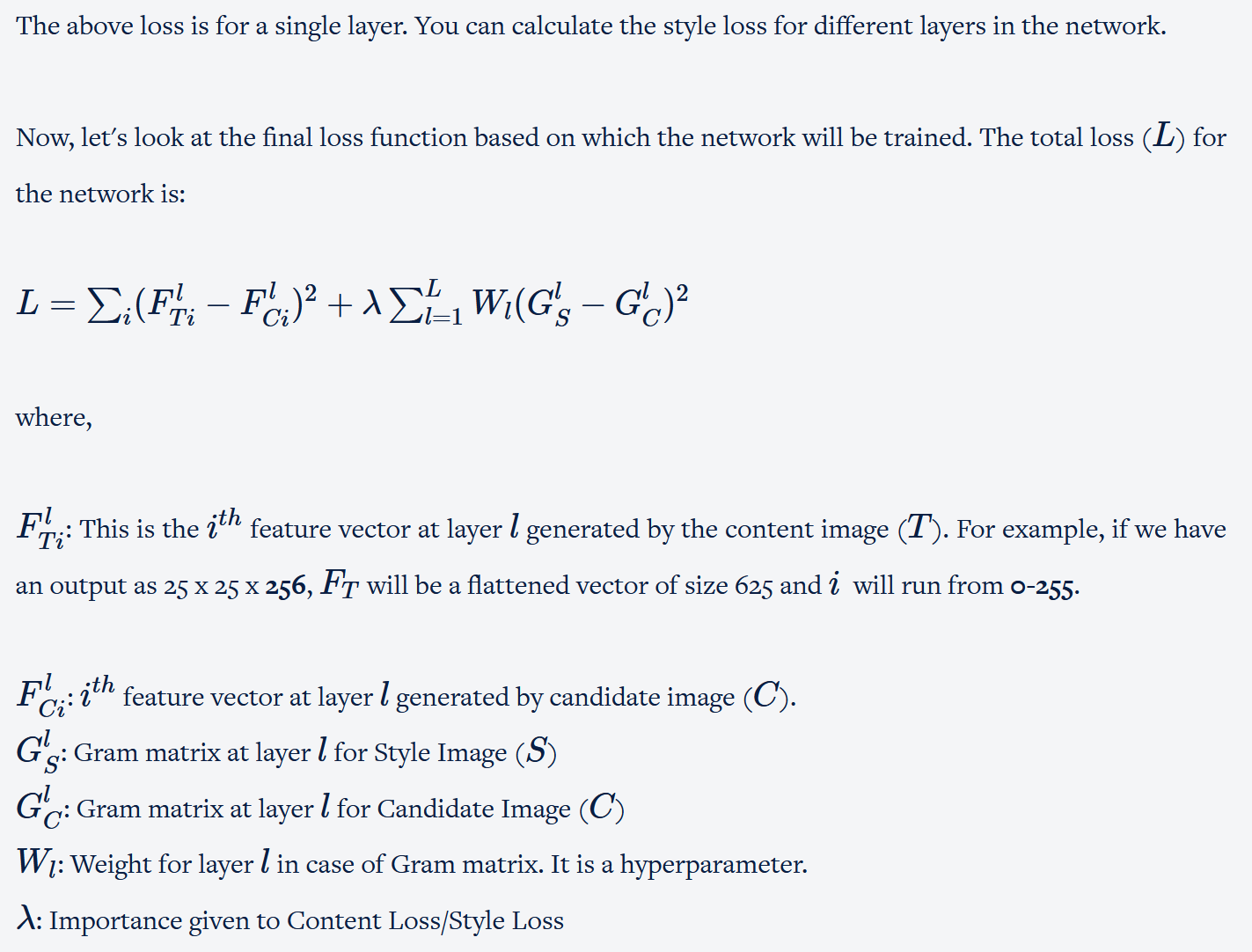
You learnt how to calculate the Gram matrix - it represents the 'style' of an image. Each element of the Gram matrix represents the correlation between two features of an image.

Recall that our goal was to generate an image - the candidate image (C) - which has a similar style as the style image (S). To tell the neural network to generate such an image we need to design a loss function that has two components - the content loss and the style loss.

In this lecture, let's look at how you can use the Gram matrix to calculate the 'style loss' between the candidate image and the style image.







Now, let's look at how the training takes place in such a network.

In the case of style transfer,  we do not change the weights of the network. Networks like VGGNet extract features from images which are used to calculate the style loss and the content loss. Since, we are training the candidate image (not the weights in the network), which has style aspect of style image and content aspect of the content image, we are minimising the total loss w.r.t to the candidate image. So, the backpropagation is on pixels of the candidate image instead of the weights of the network.

In the next segment, you will see the practical implementation of style transfer in Python.

## Style Transfer Notebook

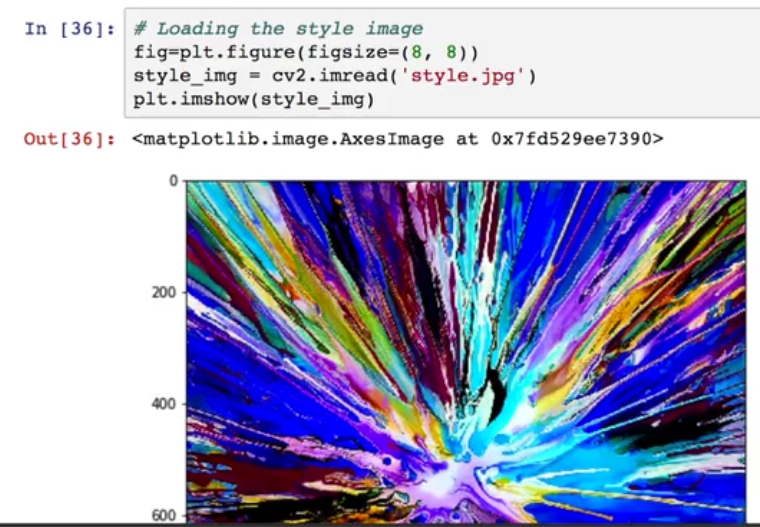
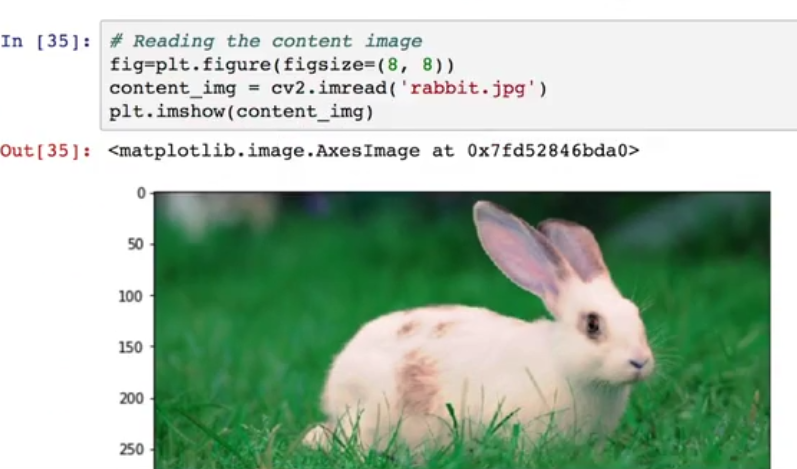
In this segment, you will see the python implementation of style transfer. For the demonstration, let’s take the image of a rabbit as the content image and splash of colours as the style image. You can, of course, use any set of images for your notebook.

Please note that there is a slight change in notation. The candidate image is denoted by ‘generated image’ in the notebook.

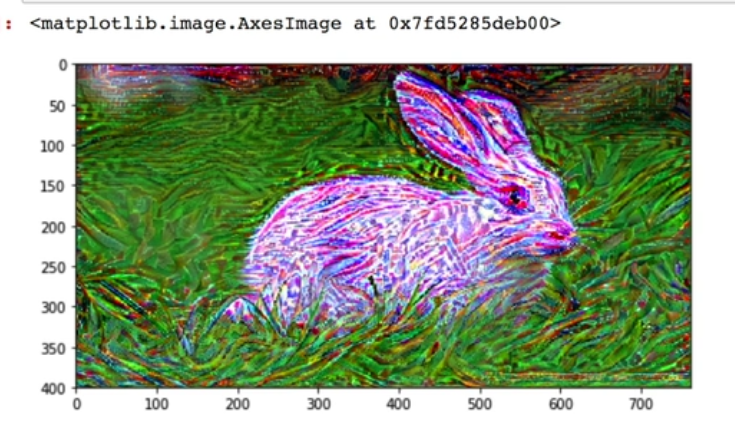
In the case of transfer learning, we use a pre-trained model, VGGNet in our case, and remove its last fully connected layer since our task is not to classify the image. Here, we use the pre-trained weights of the model to update the candidate image instead of updating the model weights. Since there are no prebuilt functions to train these type of networks in Keras,  you will see that we will use lots of custom functions.

It is highly recommended to download and run the code while going through the video.

**NOTE: The code in all of the notebooks provided for this session is updated occasionally. Therefore it may vary slightly from the one shown in the videos.**



You learnt how to load the VGG-19 model, preprocess the data, and then define the placeholder for the generated image.  Next, we will define the loss functions, define the layer from which we want to extract the features, define custom keras function and train the pixels of the generated image.

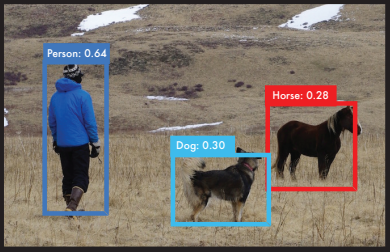


We have trained for 10 iterations. You can see the difference between the generated image at iteration-0 and iteration-5 and iteration-9. As the number of iterations increases, there is a clear effect of the style image on the generated image.

In the next few segments, you'll learn another important application of CNNs -**object detection**.

## Object Detection - I

When we were discussing the applications of CNNs, we discussed an example of object detection, which is basically the task of locating objects in an image. For example, take a look at the following image where three objects are detected. Each detected object has an associated probability or confidence.

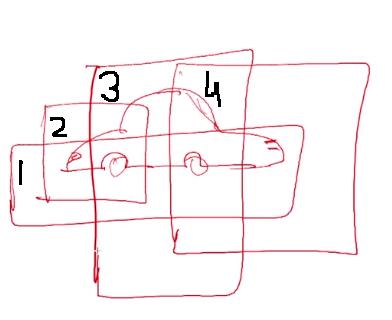


Object Detection

Let's go through the basic concept of object detection in the following lecture.

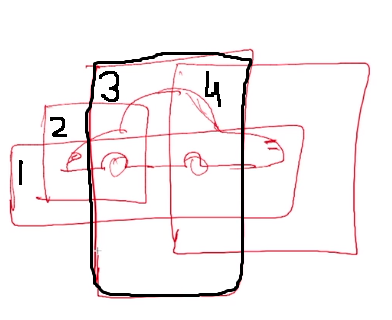
To summarise, an object detection problem can be divided into 3 subtasks:

* **Region proposal generation**: Here, we find the possible areas/regions, their shapes and sizes, which potentially contain an object. This task is where most object detection algorithms try to find novel ideas to improve state-of-the-art object detectors. You saw the example of region proposals of a car. You could have region proposals such as the one shown in the image below.



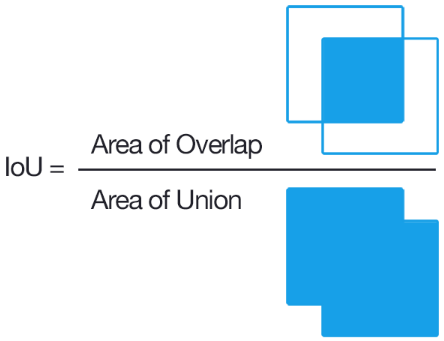
Region Proposal

* **Object classification**: The next task is to feed all the regions to an image classifier which is a CNN network.  The CNN outputs a class along with a probability of the class for every region. The regions with a probability less than a threshold value are discarded.
* **Non-maximum suppression**: There is a negligible chance that a region generated in the first step will capture the object entirely. Each region will contain a part of the object. In the end, we just want one region, which captures the object entirely. This is where the non-maximum suppression technique is used. Consider the four region proposals of the car, each of which is classified as containing a car above the stipulated threshold. (Please note that the professor had mistakenly referred to this technique as maximum suppression instead of non-maximum suppression).



Non-max Supression

In the above picture, you can see that only the 3rd box captures most of the car and other boxes are redundant.  We will do non-max suppression to remove those by calculating the metric**- IoU (intersection over union).** It is the ratio of the area of intersection over the area of union between two given regions.



IoU

We find the IoU metric between the **ground truth region of interest** and **all the generated regions of interests**. The ground truth region of interest is the ideal bounding box around the car which is present in the label in the training data. The algorithm will find that that the IoU is maximum for the 3rd box as it is closest to the ideal bounding box of the car. We will remove (or compress) the other 3 boxes.

In the figure below, you can see the various region proposals and the ground truth bounding box.



Bounding Box

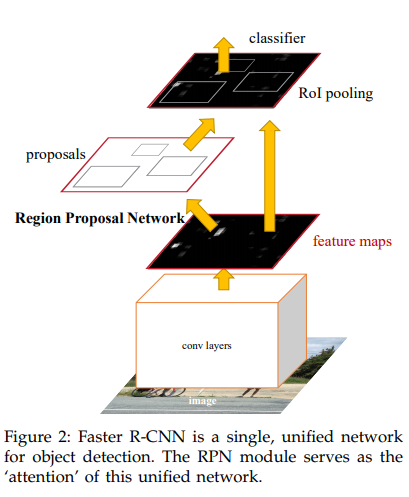
In the next segment, we will discuss another paper: 'Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Network' by S Ren. It's an improvement over the R-CNN network.

## Object Detection - II

In this segment, we will discuss the famous paper for object detection - 'Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Network' by S Ren which appeared in 2017.  It is highly recommended to go through the [paper](https://arxiv.org/pdf/1506.01497.pdf) before watching the lecture.

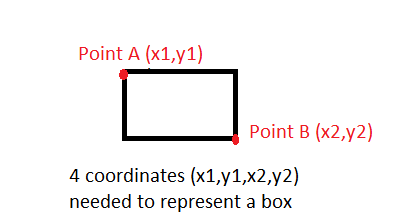
The architecture of **Faster R-CNN**is divided into the following subtasks:

* **Feature extraction**: An image is fed to the network. The initial convolutional layers generate feature maps for the image.



Faster RCNN

* **Region Proposal Network (RPN):** The feature maps are fed to the RPN which is also a part of the entire network. The RPN uses to feature maps to generate regions of interest (RoI). Unlike the R-CNN network which randomly generated the RoIs, The Faster R-CNN generates RoIs smartly. The region generation or region proposal is also learnt while training the network. The RPN scans the feature maps generated in the previous step to find potential RoIs. After classifies a RoI, an RPN generates two things about the detected RoI. The **first** thing that it generates is an **objectness score**which is a probability of the RoI box having an object or not. For example, the objectness score could be 0.83 which tells us that the detected RoI has a high probability of having an object inside it. It doesn't classify the object yet, it just tells whether it contains an object or not. The **second** thing generated by the RPN is a **pair of coordinates** that tell the location of the RoI box in the image. For example, the RNP can generate a pair of coordinates as shown below. The coordinates are diagonally opposite which can uniquely identify a box.



Box

* Next, we give the **output of RPN** and the **features extracted by the initial convolutional layers** to the **classifier** present at the end of the network. We push the output of RPN to the classifier only if the objectness score is above a certain threshold such as 0.5 which means the RPN thinks that there is a high probability of the presence of an object inside the RoI box.
* The classifier performs two tasks. First, it **classifies** the object present in the RoI box into one of the classes such as car, cat, dog, etc. It then uses the **log loss**by comparing the classified object with the ground truth object. The second task is to calculate the **regression loss** which compares the 4 coordinates generated by the RPN with the ground truth bounding box coordinates. Both of these losses are backpropagated to update the weights in the entire network including those of the RPN.

## Summary

In this session, you learnt about Style Transfer and Object Detection. In style transfer, you learnt the meaning of style and content in an image, combining those in varying ratio to give candidate image. Also, we used pre-trained weights to extract the features from style, content and candidate image to compute the loss. We had not changed the pre-trained weights but took the gradient of loss w.r.t to the pixels of the candidate image.

In Object Detection, you learnt about Region Proposal, Object Classification and Non-maximum suppression. In the Region Proposal, we find all potential areas/regions that can contain an object. In the Object Classification, we classify the object, and in the Non-maximum suppression, we only choose the bounding box that has maximum overlap the true bounding box and suppress the rest.