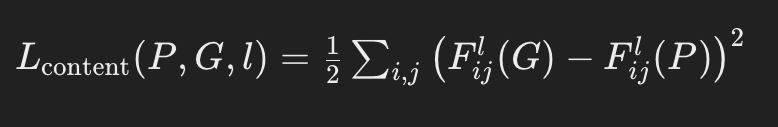
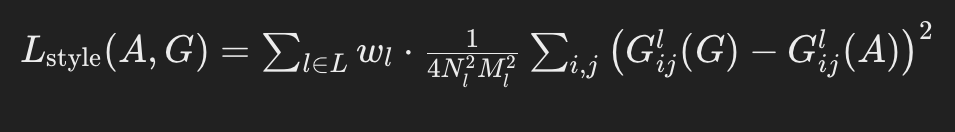
**REVIEW QUESTIONS**

**\*α & β hyperparamters**

α: The weight for the content loss. A higher value of α makes the generated image more similar to the content image in terms of structure and overall composition

β: The weight for the style loss. A higher value of β makes the generated image more similar to the style image in terms of textures, colors, and patterns.



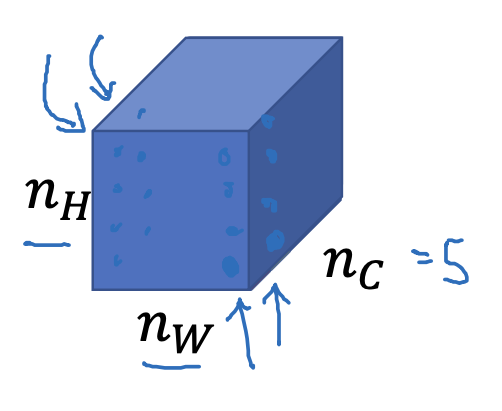
When loss function is calculated using MSE we see the above math

ematical formula for them.

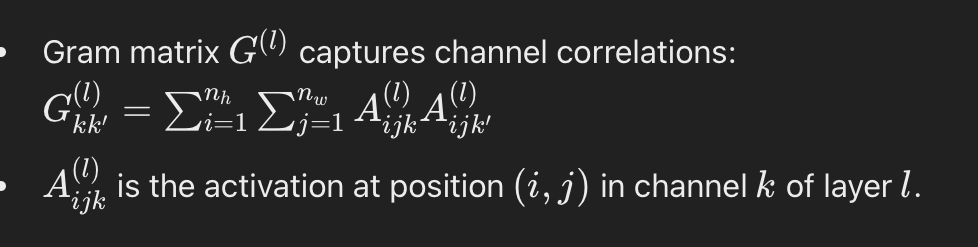
**\*Style cost function**

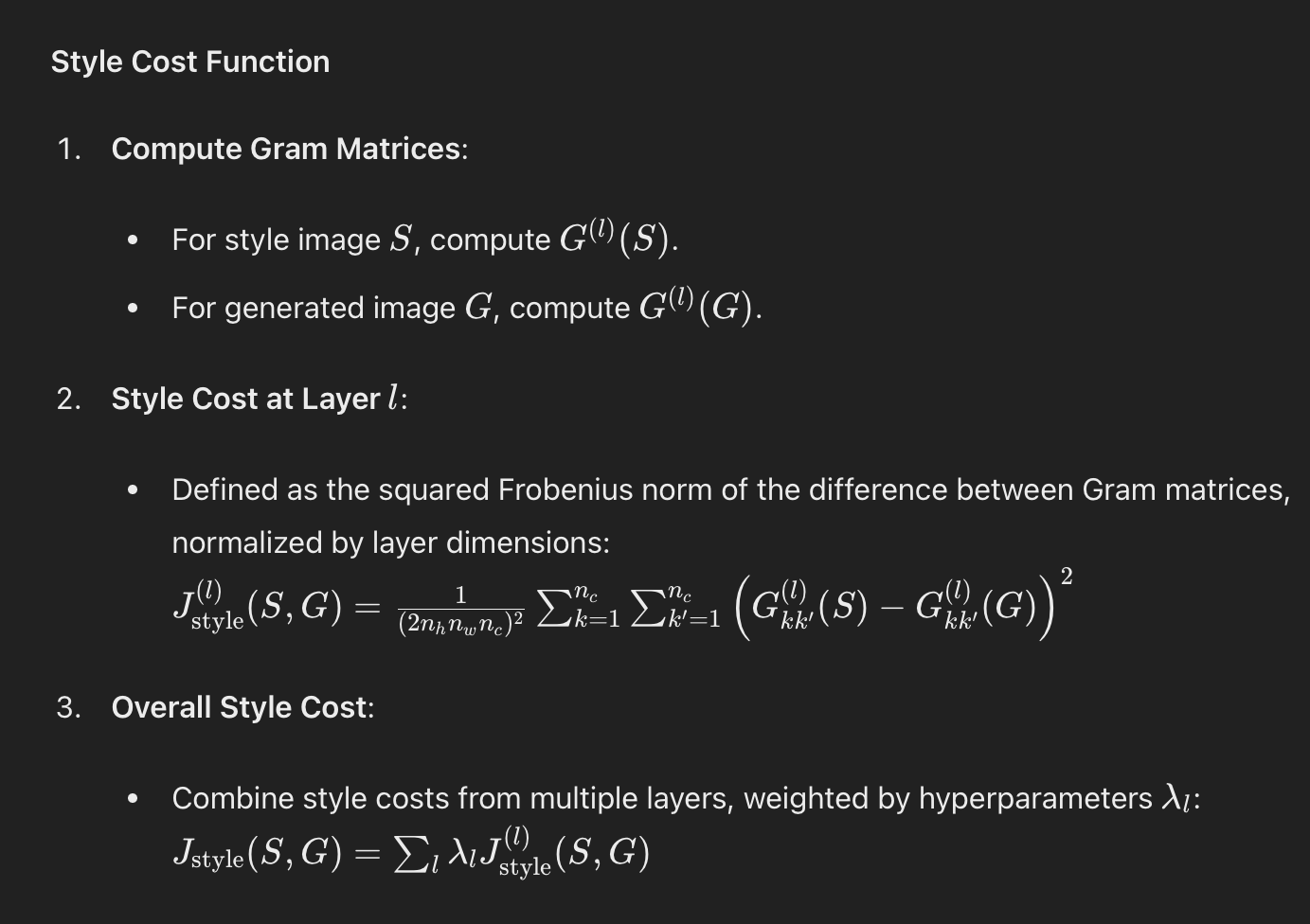
we are using layer l’s activation to measure “style.”

Define style as correlation between activations across channels.

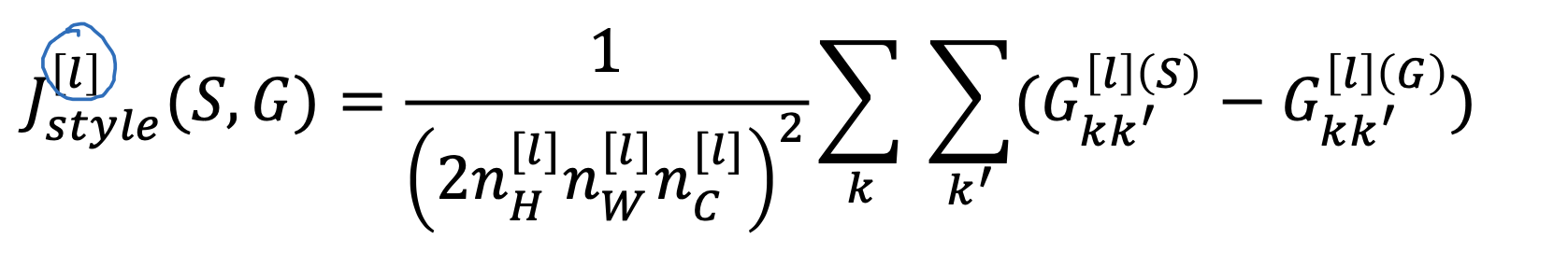
Activations form a 3D block as given below

Style is captured by correlation b/w feature channels

Style matrix -



Overall style cost function—>



**\*multi-class and multi-label**

**\*multi-class**

With multiclass classification, the model will select just one winning label. A classifier that identifies car makes, for instance, will only output one label.

Here each instance is assigned one label.

Used in MNIST digit recognition

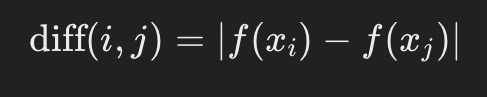
**\*multi-label**

With multilabel classification, the model has the option of outputting multiple labels if relevant.

Here one instance can be assigned with multiple labels.

Used in tagging images

**\*Binary classification for face recognition using siamese net**

We compute difference vector:

Feed the difference vector into logistic regression unit

Logistic regression unit unit applies a sigmoid function to difference vector as follows-

Create a dataset of image pairs labeled 1 if they are the same person and 0 if different. Use backprop to train the Siamese Network and logistic regression together with binary cross-entropy loss((y \* log(p) + (1 - y) \* log(1 - p)) ).

**\*Contrastive Loss function**

It is use to learn similarity or dissimilarity in siamese network.

Basically Is brings similar inputs together and pushes dissimilar labels away.

Its formula is-

L = (1 — y) \* D² + y \* max(0, m — D)²

Here,

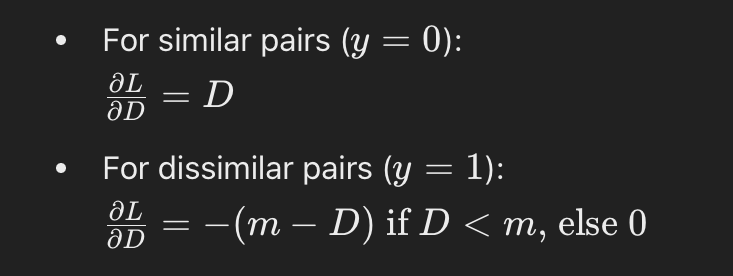
D= distance between encodings

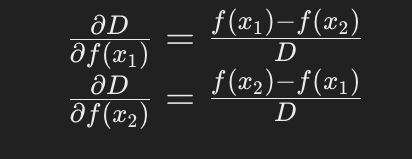
y :label indicating whether pair is similar or not

m=margin parameter which is the threshold for dissimilarity

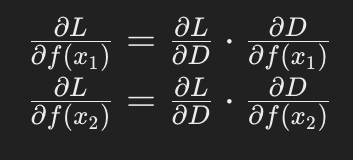
**\*Backprop in siamese net**

During bacprop network updates weights based on gradients of loss function

Gradients -

We can get the above by differentiating contrastive loss function

——>wrt feature vector



——> chain rule

Now gradients of feature vectors f(x1) & f(x2) are back propagated in the network since network contains the parameters as weights so gradient updates the weights.

The weights can be updated by using optimizing algorithm like Adam.

**\*CNN implementation on digit recognition in MNIST dataset**

Git repository -<https://github.com/rampofin/mrm.git>

1.**Libraries** used:

-torch

-torchvision

-matplotlib.pyplot

2.**Data loading**

train\_data, test\_data, val\_data = loads MNIST data set

train\_dl, val\_dl, text\_dl= creates data loaders

3.**DataLoader dictionary**

Has dictionary named data loaders for training, testing &

validation with batch size of 100.

4.**CNN class**

This defines the architecture of CNN

Conv1,conv2- convolutional layers

Conv2\_drop-dropout layer(prevents overfitting)

Fc1,fc2-fully connected layers

Forward function - defines forward pass

5.**model** - creates instance of CNN model

6.**optimizer** - Adam

7.**loss\_fn**- defines CrossEntropyLoss

8.**train(epoch)-**Handles the training process for an epoch

9.**test()-**Evaluates model performance on test dataset

10.i**nference()**- performs prediction