MATHEMATICAL MODEL OF GAN

Author1 ,*Author 2
UCD
**@ucdavis.edu{, **}@ucdavis.edu

1 Introduction

Recent paper ([4]) characterize the information after deep learning network by analyzing the eigenvalue of gram matrix after non linear neural network. They pass the Gaussian standard noise into a trained GAN model to generate three classes of images, then passing the generated data into common trained deep representation model CNN that is for multi-classification. They extract the output, which has large dimension, from one inner layer, and analyzed eigenvalue of them, and found that they have same behavior as the GMM data that was constructed by using the mean and variance of the GAN data. The paper consider the and the serials of neural network models are Lipschitz operations where the concentrations of the data are maintained after such operation. They concludes that given a number of the standard Gaussian vectors which are concentrated vectors, under large n, p domain where $n, p \to \infty$ but $\frac{n}{p} \to c < \infty$, after series Lipschitz operations, the resulting matrix remain the Lipschitz concentration.

However, the following questions are unclear.

One is the relationship between the above behavior of the representation with the parameters of GAN model. The paper generates such representation starting with the standard normal Gaussian noise vector as the concentrated vector, then passed the trained generator, to obtain the representation. Due to the concentration property of standard Gaussian and Lipschitz properties of operations in GAN, the resulting Gram matrix has its first and second order statistics same with the GMM. However, such parameters of GAN may not be necessary for deducing the result.

The other unclear question is whether the distribution of noise vector matters. The standard normal Gaussian is the concentrated vector, what if we change the distribution of the noise vector, does the result still hold?

Besides answering the above two questions, we also include an additional simulation environment, the MNIST ([**?**]) classification task, with more simpler model.

2 Review of GAN

Since the birth of deep learning, it has been widely used in various fields, such as robot control, human voice authentication, biological network analysis, and image generation. In recent years, part of scientific research has focused on how to better

*Preprint. Work in progress

explain the deep learning model. Differentiable Generative network is one of the neural networks that transfers a latent variable z to a sample x or the distribution of x by the differentiable neural network. Generative Adversarial Network (GAN) ([3]) is belong to the differentiable generative network and is based on the game theoretical scenario where the generator is responsible for generating the training data $x^{(g)} = g(z, \theta^{(g)})$, and at the same time the discriminator is responsible for discriminating the real and generated data by maintaining a probability of the data being as real $d(x; \theta^d)$. A payoff function

$$c(\theta^g, \theta^d) = \mathbb{E}_{x^r} log d(x^r) + \mathbb{E}_{x \sim g} log(1 - d(x^g))$$

is defined to for the discriminator so that it can better distinguish the x^r and x^g. The generator is trained upon an approach by the zero sum mean game such that

$$g^* = \underset{g}{\operatorname{argmin}}\, max_d c(g, d)$$

. The conditional GAN is an extension of transitional GAN such that the overall model contains multiple GAN and each one generates data of a specific class conditionally on a class label.

Random Matrix Theory With GAN Data Their noise vector is passed through the BigGAN model ([1]) that was pre-trained from the Imagenet dataset ([2]) then followed with some popular CNN networks (Resnet50, VGG16, and Densenet201). The paper GAN-data generation and representation process is a series of Lipschitz transformations such that the concentrated vectors are stable after Lipschitz transformation. The paper also gave a proof of Lipschitz constant control over the neural network with affine layers with spectral normalization. The paper also proved that the Gram matrix and its estimation after deep learning representation is well bounded in large dimension domain with some assumptions.

3 Mathmatical Model

3.1 Embedding and Conditional Random Noise Vector

Given a random noise vector $z \in \mathbb{R}^t$, and a fixed label as $l \in \mathbb{R}^{|L|}$, where t is a parameter, for example we can choose $d = 128$ by convention. l is a one hot vector to denote the class label, and L is a set of all labels. For example, we can let $|L| = 1000$. Then a given an embedding function is defined as

$$E : lA^T \to e \in \mathbb{R}^t \tag{1}$$

where $A \in \mathbb{R}^{t \times |L|}$ is a weight matrix that is initialized from $\mathcal{N}(0, 1)$. We then create a conditional vector by row-wised stacking of z and e.

$$c = (z^T, e^T) \in \mathbb{R}^d \tag{2}$$

where $d = 2t$. For future reference, we use d_7 to denote $d = 2^7 = 128$, and we use $d_8, d_9, d_{10}, d_{11}, d_{12}$ to denote $2^8 = 256, 2^9 = 512, 2^{10} = 1024, 2^{11} = 2048, 2^{12} = 4096$.

The generator is a sequential functions that applied to c and we will give their formulations as below.

3.2 Linear Transformation with Spectral Normalization

A **linear transformation**

$$\mathcal{F} : cA^T + b \to x \in \mathbb{R}^p \tag{3}$$

where $A \in \mathbb{R}^{p \times d}$ is a fixed weight matrix after training, and $b \in \mathbb{R}^p$ is fixed vector of bias, which was originally randomly initialized. We can choose $p = 32678, d = 256$.

Spectral Normalization The Spectral normalization is a numerical operation applied to the A. Given the initialized raw $A_o \in \mathbb{R}^{p \times d}$ and two random vectors $u \in \mathbb{R}^p$ and $v \in \mathbb{R}^d$, the following operations are performed on A_o.

$$A = \frac{A_o}{\sigma^A} \tag{4}$$

where

$$\begin{aligned} \sigma^A &= u' \cdot (A_o v') \\ v' &= \frac{uA_o}{\max(\|uA_o\|_p, \epsilon)} \\ u' &= \frac{A_o v}{\max(\|A_o v\|_p, \epsilon)} \end{aligned}$$

Refactoring Given the previous output x after , then we refactor it to a tensor in$\mathbb{R}^{1 \times H \times W \times C}$ then permute it as $x \in \mathbb{R}^{1 \times C \times H \times W}$. Where W, H are parameters, we can choose $W = H = 4$. Then it's easy to calculate that $C = d_{11} = 2048$ in our example.

This x will be passing through series of sequential layers.

3.3 Sequential Layers

Given such x and c, the next steps are sequential layers that applied to them.

$$\mathcal{L}_n \circ \cdots \circ \mathcal{L}_0(x, c, \sigma) \tag{5}$$

Where $x \in \mathbb{R}^{1 \times H \times W \times C} := \mathcal{F}(c, A, b)$. Here the $\mathcal{L}$ could be one of any module structures as described below. Here we introduced four different modules, and the sequential layers is composed of n modules from these four categories. We also introduce another constant $\sigma \in (0, 1)$, which is also a parameter.

3.3.1 Generator Block Module Without Up Sampling and Without Drop Channels

$\mathcal{L}_0$

Batch Normalization - $\mathcal{B}_0$ Here we define some pre-computed statistics for this operation. Let $f^{(B_0)}$, $s^{(B_0)}$ are the fractional and integer parts of $\frac{\sigma}{\eta}$. Let $r^{(B_0)}_{means} \in 0^{r_1^{(B_0)} \times C}$ and $r^{(B_0)}_{vars} \in 1^{r_1^{(B_0)} \times C}$, and $\eta = \frac{1}{r_1^{(B_0)} - r_2^{(B_0)}}$. And $r^{(B_0)}_{mean} = (r^{(B_0)}_{means})_{s^{(B_0)},:}$, $r^{(B_0)}_{var} = (r^{(B_0)}_{vars})_{s^{(B_0)},:}$ are sliced vectors from the matrix by setting the row index

equals to $s^{(B_0)}$. Then we refactor $r_{mean}^{(B_0)} \in 0^{1\times C\times 1\times 1}$, $r_{var}^{(B_0)} \in 1^{1\times C\times 1\times 1}$. We set hyperparameter $r_1^{(B_0)} = 51, r_2^{(B_0)} = 1$.

Then we create weight w and bias b by following operations: Here we let $A^{(B_0)} \in \mathbb{R}^{C\times d}$ is a spectral Normalized weight matrix. Then we apply a linear operation to c such that

$$w \in \mathbb{R}^C := 1 + cA^{(B_0)T} \tag{6}$$

and we refactor $w \in \mathbb{R}^{C\times 1\times 1}$ and

$$b \in \mathbb{R}^{1\times C} : cA^{(B_0)T} \tag{7}$$

and we refactor $b \in \mathbb{R}^{1\times C\times 1\times 1}$. So a batch normalization $\mathcal{B}_0$ is defined as

$$\mathcal{B}_0 : \frac{x - r_{mean}}{\sqrt{r_{var} + 0.0001}} * w + b \to x^{(B_0)} \in \mathbb{R}^{1\times C\times W\times H} \tag{8}$$

RelU - $\mathcal{R}_0$ The rectified liner unit function is applied afterwards such that

$$R_0(x^{(B_0)}) : max(0, x^{(B_0)}) \to x^{(R_0)} \tag{9}$$

Convolution - C_0 We let filter $A^{(C_0)}$ as the weight matrix after spectral normalization, which is a tensor in $\mathbb{R}^{C^{(C_0)}\times \frac{C}{groups}\times kH\times kW}$, and bias $b \in \mathbb{R}^{C^{(C_0)}}$, where $C^{(C_0)}, kH, kW, groups$ are parameters.

The convolution $\mathcal{C}_0$ is mapping from input $x^{(R_0)} \in \mathcal{R}^{N\times C\times H\times W}$ to $x^{(C_0)} \in \mathbb{R}^{N\times C^{(C_0)}\times H^{(C_0)}\times W^{(C_0)}}$, which is defined as:

$$\mathcal{C}_0(x^{(R_0)}) \to x^{(C_0)} \in \mathbb{R}^{1\times C^{(C_0)}\times H^{(C_0)}\times W^{(C_0)}} \tag{10}$$

where

$$H^{(C_0)} = \frac{H + 2\times padH - dH\times(kH-1) - 1}{sH} + 1 \tag{11}$$

$$W^{(C_0)} = \frac{W + 2\times padW - dW\times(kW-1) - 1}{sW} + 1 \tag{12}$$

For example, we have $C^{(C_0)} = d_4 = 512, C_{in} = d_{16} = 2048, groups = 1$, we also add following parameters: let $iH = 4, iW = 4, kH = 1, kW = 1, sH = 1, sW = 1, dH = 1, dW = 1$, it means $stride = 1$, and $padding = 0$, and $dilation = 1$. Then the $H^{(C_0)} = W^{(C_0)} = 4$

Batch Normalization $\mathcal{B}_1$ and Relu R_1 It has the same operations as $\mathcal{B}_0$, so that

$$x^{(B_1)} \in \mathbb{R}^{1\times C^{(C_0)}\times H^{(C_0)}\times W^{(C_0)}} = \mathcal{B}_1(x^{(C_0)}, c, \sigma) \tag{13}$$

$$x^{(R_1)} \in \mathbb{R}^{1\times C^{(C_0)}\times H^{(C_0)}\times W^{(C_0)}} = R_1(x^{(B_1)}) = \max(0, x^{B_1}) \tag{14}$$

where $C_{(C_0)} = d_4 = 512$

Convolution $\mathcal{C}_1$ We let filter $A^{(C_1)}$ as the weight matrix after spectral normalization, which is a tensor in $\mathbb{R}^{C^{(C_1)} \times \frac{C^{(C_0)}}{groups} \times kH \times kW}$, and let bias as $b \in \mathbb{R}^{C^{(C_1)}}$,

The $\mathcal{C}_1$ is a mapping from input $x^{(R_1)} \in \mathbb{R}^{N \times C^{(C_0)} \times H^{(C_0)} \times W^{(C_0)}}$ to $x^{(C_1)} \in \mathbb{R}^{N \times C^{(C_1)} \times H^{(C_1)} \times W^{(C_1)}}$ defined as:

$$\mathcal{C}_1(x^{(R_1)}) \to x^{(C_1)} \in \mathbb{R}^{1 \times C^{(C_1)} \times H^{(C_1)} \times W^{(C_1)}} \tag{15}$$

here we let $A^{(C_1)} \in \mathbb{R}^{512 \times 512 \times 3 \times 3}$ as, we let $b^{(C_1)} \in \mathbb{R}^{512}$. So $C_{out} = d_4 = 512, C_{in} = d_4 = 512$. For our calculation, we also add following variables, $iH = 4, iW = 4, kH = 1, kW = 1, sH = 1, sW = 1, dH = 1, dW = 1, padH = 1, padW = 1$, it means $kernelsize = 3$, $stride = 1$, and $padding = 1$, and $dilation = 1$, and $groups = 1$. Then the $H^{(C_1)} = W^{(C_1)} = 4$

Batch Normalization $\mathcal{B}_2$ and ReLU $\mathcal{R}_2$ and Convolution $\mathcal{C}_2$ It has the same operations as $\mathcal{B}_0, \mathcal{B}_1$, so that

$$x^{(B_2)} \in \mathbb{R}^{1 \times C^{(C_1)} \times H^{(C_1)} \times W^{(C_1)}} := \mathcal{B}_2(x^{(C_1)}, c, \sigma) \tag{16}$$

$$x^{(R_2)} := R_2(x^{(B_2)}) \max(0, x) \tag{17}$$

$$\mathcal{C}_2(x^{(R_2)}) \to x^{(C_2)} \in \mathbb{R}^{1 \times C^{(C_2)} \times H^{(C_2)} \times W^{(C_2)}} \tag{18}$$

where $C^{(C_2)} = d_9 = 512, H^{(C_2)} = W^{(C_2)} = 4$

Batch Normalization $\mathcal{B}_3$ and ReLU $\mathcal{R}_3$ It has the same operations as $\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2$, so that

$$x^{(B_3)} \in \mathbb{R}^{1 \times C^{(C_2)} \times H^{(C_2)} \times W^{(C_2)}} := \mathcal{B}_3(x^{(C_2)}, c, \sigma) \tag{19}$$

$$x^{(R_3)} := ReLU(x^{(B_3)}) = \max(0, x^{(B_3)}) \tag{20}$$

Convolution - $\mathcal{C}_3$ Let filter $A^{(C_3)}$ as the weight matrix after spectral normalization, a tensor in $\mathbb{R}^{C^{C_3} \times \frac{C^{C_2}}{groups} \times kH \times kW}$, And bias $b \in \mathbb{R}^{C^{C_3}}$,

The convolution - 3 $\mathcal{C}_3$ is mapping from input $x^{(C_2)} \in \mathbb{R}^{N \times C^{(C_2)} \times H^{(C_2)} \times W^{(C_2)}}$ to $x^{(C_3)} \in \mathbb{R}^{N \times C^{(C_3)} \times H^{(C_3)} \times W^{(C_3)}}$ defined as:

$$\mathcal{C}_3(x^{(R_3)}) \to x^{(C_3)} \in \mathbb{R}^{1 \times C^{C_3} \times H^{(C_3)} \times W^{(C_3)}} \tag{21}$$

here we let $A \in \mathbb{R}^{2048 \times 512 \times 1 \times 1}$, we let $b \in \mathbb{R}^{2048}$. Here we let $C^{(C_3)} = d_{11} = 2048$, and $C^{(C_2)} = d_9 = 512$. For our calculation, we also add following variable $H = 1, W = 1$, $H^{(C_2)} = 4, W^{(C_2)} = 4, kH = 1, kW = 1, sH = 1, sW = 1, dH = 1, dW = 1, padH = 0, padW = 0$, it means $kernelsize = 1$, $stride = 1$, and $padding = 1$, and $dilation = 1$, and $groups = 1$. And $H^{(C_3)} = W^{(C_3)} = 4$

Addition - $\mathcal{A}_0$ The addition operation is

$$x^{(C_3)} + x \to x^{(A_0)} \in \mathbb{R}^{1 \times C^{(C_3)} \times H^{(C_3)} \times W^{(C_3)}} \tag{22}$$

3.3.2 Generator Block Module With Up Sampling and Without Drop Channels

This block module has one more operation up-sampling. Here we simplify the description of the model since it's the same above. $\mathcal{L}_1$ is a such module.

Batch Normalization - 0, ReLU, Convolution - 0, Batch Normalization - 1, RelU This series functions generate the $x^{(L_1,R_1)} \in \mathbb{R}^{1\times C^{(L_1,C_0)}\times H^{(L_1,C_0)}\times W^{(L_1,C_0)}}$ from input $x^{(L_0,A_0)} \in \mathbb{R}^{1\times C^{(L_0,C_3)}\times 4\times 4}$, where $C_0 = d_9 = 512$, and $H^{(L_1,C_0)} = W^{(L_1,C_0)} = 4$.

Up Sampling Up sampling is a interpolate function. Here we use the nearest neighbor in the last two dimension to interpolate the values. so:

$$\mathcal{U}(x^{(L_1,R_1)}) :\rightarrow x^{(L_1,U_0)} \in \mathbb{R}^{1\times C^{(L_1,C_0)}\times H^{(L_1,U)}\times W^{(L_1,U)}} \tag{23}$$

where $H^{(L_1,U)} = W^{(L_1,U)} = 8$

Convolution - 1, Batch Normalization - 2, Relu, Convolution - 2, Batch Normalization - 3, Relu, Convolution - 3 The convolution - 1 to convolution - 3 have the same definitions as before. And before convolution - 3, the $x \in \mathbb{R}^{1\times C^{(L_1,C_2)}\times H^{(L_1,C_2)}\times W^{L_1,C_2}}$, where $C^{(L_1,C_2)} = d_9 = 512$ and $H^{(L_1,C_2)} = W^{(L_1,C_2)} = 8$. The convolution - 3 outputs $x^{(L_1,C_3)} \in \mathbb{R}^{1\times C^{(L_1,C_3)}\times H^{(L_1,C_3)}\times W^{(L_1,C_3)}}$, where $C^{(L_1,C_3)} = 2048$ and $H^{(L_1,C_3)} = W^{(L_1,C_3)} = 8$.

Up Sampling to x_o Here the same up-sampling procedure applied to $x^{(L_0,A_0)}$ such that:

$$\mathcal{U}(x^{(L_0,A_0)}) :\rightarrow x^{(L_1,U_1)} \in \mathbb{R}^{1\times C^{(L_1,U_1)}\times H^{(L_1,U_1)}\times W^{(L_1,U_1)}} \tag{24}$$

where $C^{(L_1,U_1)} = d_{11} = 2048$ and $H^{(L_1,U_1)} = W^{(L_1,U_1)} = 8$

Addition The last layer is addition operation same as previous

$$x^{(L_1,C_3)} + x^{(L_1,U_1)} \rightarrow x \in \mathbb{R}^{1\times C^{(L_1,C_3)}\times H^{(L_1,C_3)}\times W^{(L_1,C_3)}} \tag{25}$$

3.3.3 Generator Block Module with Up Sampling and With Drop Channels

Batch normalization - 0, Relu, Convolution - 0, Batch Normalization -1, Relu, Upsampling, Convolution - 1, Batch Normalization -2, Relu, Convolution -2, Batch Normalization -3, Relu The generator block module with up sampling and with drop channels has the same above operations as the generator block module with up sampling and without drop channels. $\mathcal{L}_3$ is a such module. After previous operations up to $\mathcal{L}_2$, the $x^{(L_3,A_0)} \in \mathbb{R}^{1\times C^{(L_2,C_3)}\times H^{(L_2,C_3)}\times W^{(L_2,C_3)}}$, where $C^{(L_2,C_3)} = d_{11} = 2048$ and $H^{(L_2,C_3)} = W^{(L_2,C_3)} = 8$

Convolution - 3 Let filter A as the weight matrix after spectral normalization , a tensor in $\mathbb{R}^{C^{(L_3,C_3)}\times \frac{C^{L_3,C_2}}{groups}\times kH\times kW}$. And bias $b \in \mathbb{R}^{C_{out}}$. we let $A \in \mathbb{R}^{1024\times 512\times 1\times 1}$, $b \in \mathbb{R}^{1024}$. Here we let $C_{out} = d_{10} = 1024$, and $C_{in} = d_9 = 512$. For our calculation, we also add following variable $H = 1, W = 1$, and the input height and weight can be: $iH = 16, iW = 16$, the kernel height and weight can be $kH = 1, kW = 1, sH = 1, sW = 1, dH = 1, dW = 1, padH = 0, padW = 0$, it means $kernelsize = 1$, $stride = 1$, and $padding = 1$, and $dilation = 1$, and $groups = 1$.

The convolution $\mathcal{C}_3$ is mapping from input $x \in \mathbb{R}^{N\times C_{in}\times H^{(L_3,C_2)}\times W^{(L_3,C_2)}}$ to $x \in \mathbb{R}^{N\times C^{(L_3,C_3)}\times H^{(L_3,C_3)}\times W^{(L_3,C_3)}}$ defined as:

$$\mathcal{C}_3(x) \rightarrow x^{(L_3,C_3)} \in \mathbb{R}^{1\times C_{out}\times H\times W} \tag{26}$$

where $H^{(L_3,C_3)} = W^{(L_3,C_3)} = 16$, and $C^{(L_3,C_3)} = d_{10} = 1024$

Drop Channel Since the $x^{(L_2,A_0)}$ and $x^{(L_3,C_3)}$ has different dimension in the second dimension output channels, here we do the drop channel such that only select the ones corresponding to the dimension of x, so that

$$\mathcal{D}(x^{(L_2,A_0)} \in \mathbb{R}^{1\times C^{(L_2,C_3)}\times H^{(L_2,C_3)}\times W^{(L_2,C_3)}}) :\to x^{(L_3,D)} \in \mathbb{R}^{1\times C^{(L_3,D)}\times H^{(L_2,C_3)}\times W^{(L_2,C_3)}} \tag{27}$$

where $C^{(L_3,D)} = 1024$

Up Sampling x_o Here the same up-sampling procedure applied to x_0 such that:

$$\mathcal{U}(x^{(L_3,D)}) :\to x^{(L_3,U_1)} \in \mathbb{R}^{1\times C^{(L_3,D)}\times H^{(L_3,U_1)}\times W^{(L_3,U_1)}} \tag{28}$$

where $H^{(L_3,U_1)} = W^{(L_3,U_1)} = 16$

Addition The last layer is addition operation same as previous

$$x^{(L_3,C_3)} + x^{(L_3,U_1)} \to x^{(L_3,A_0)} \in \mathbb{R}^{1\times C^{(L_3,C_3)}\times H^{(L_3,C_3)}\times W^{(L_3,C_3)}} \tag{29}$$

3.3.4 Self Attention Layers

$\mathcal{L}_8$ module has self attention layers. After some block modules, then the $x^{(L_7,A_0)} \in \mathbb{R}^{1\times d_9\times 64\times 64}$ will pass a self attention layers. where $C^{(L_7,C_3)} = 512$ and $H^{(L_7,C_3)} = W^{(L_7,C_3)} = 64$

Convolution - θ path Let filter A as the weight matrix after spectral normalization, which is a tensor in $\mathbb{R}^{C_{out}\times \frac{C_{in}}{groups},kH,kW}$, And the input is in $\mathbb{R}^{batch\times C_{in}\times iH\times iW}$.

The convolution $\mathcal{C}_\theta$ is mapping from input $x^{(L_7,A_0)} \in \mathbb{R}^{N\times C_{in}\times iH\times iW}$ to $\theta \in \mathbb{R}^{N\times C^{(L_8,C_\theta)}\times oH\times oW}$ defined as:

$$\mathcal{C}_\theta(x) \to \theta \in \mathbb{R}^{1\times C_{out}\times 64\times 64} \tag{30}$$

where $C^{(L_8,C_\theta)} = 64$ and $H^{(L_8,C_\theta)} = W^{(L_8,C_\theta)} = 64$

Here we let $A \in \mathbb{R}^{64\times 512\times 1\times 1}$ $C_{out} = d_6 = 64$, and $C_{in} = d_9 = 512$. height and weight can be: $iH = 64, iW = 64, C_{in} = 512$, the kernel height and weight can be $kH = 1, kW = 1, sH = 1, sW = 1, dH = 1, dW = 1, padH = 0, padW = 0$, it means $kernelsize = 1$, $stride = 1$, and $padding = 1$, and $dilation = 1$, and $groups = 1$.

Refactor We than refactor θ into a matrix such that $\theta^{(Ref)} \in \mathbb{R}^{N^{(Ref)}\times C^{(L_7,C_3)}/8\times (H^{(L_7,C_3)}\times W^{(L_7,C_3)})}$, where $N^{(Ref)} = \frac{batch\times C_{out}\times oH\times oW}{C_{in}/8\times iH\times iW}$, for example, using the above x, we get new $\theta \in \mathbb{R}^{1\times 64\times 4096}$.

Convolution - ϕ with max pooling Let filter A as the weight matrix after spectral normalization, which is a tensor in $\mathbb{R}^{C_{out}\times \frac{C_{in}}{groups}\times kH\times kW}$, And the input is in $\mathbb{R}^{batch\times C_{in}\times iH\times iW}$.

The convolution $\mathcal{C}_\phi$ is mapping from input $x^{(L_7,A_0)} \in \mathbb{R}^{N\times C_{in}\times iH\times iW}$ to $\phi \in \mathbb{R}^{N\times C^{(L_8,C_\phi)}\times oH\times oW}$ defined as:

$$\mathcal{C}_\phi(x) \to \phi^{(L_8,C_\phi)} \in \mathbb{R}^{1\times C_{out}\times 64\times 64} \tag{31}$$

where $C^{(L_8,C_\phi)} = 64$ and $H^{(L_8,C_\phi)} = W^{(L_8,C_\phi)} = 64$

Here we let here we let $A \in \mathbb{R}^{64\times512\times1\times1}$ and $C_{out} = d_8 = 64$, and $C_{in} = d_9 = 512$. The height and weight can be: $iH = 64, iW = 64, C_{in} = 512$, the kernel height and weight can be $kH = 1, kW = 1, sH = 1, sW = 1, dH = 1, dW = 1, padH = 0, padW = 0$, it means $kernelsize = 1$, $stride = 1$, and $padding = 1$, and $dilation = 1$, and $groups = 1$.

A max pooling layer is followed after the $\mathcal{C}_\phi(x)$, Here we have input $\phi \in \mathbb{R}^{batch\times C_{in}\times iH\times iW}$. A max pooling layer is function mapping from input $\theta \in \mathbb{R}^{N\times C_{in}\times iH\times iW}$ to $\theta \in \mathbb{R}^{N\times C_{out}\times oH\times oW}$ defined as:

$$Maxpooling_\phi(\phi) \to \phi^{(L_8,Mp_0)} \in \mathbb{R}^{1\times64\times32\times32} \tag{32}$$

where

$$oH = \frac{iH + 2\times padH - dH\times(kH-1) - 1}{sH} + 1 \tag{33}$$

$$oW = \frac{iW + 2\times padW - dW\times(kW-1) - 1}{sW} + 1 \tag{34}$$

Here we let $C_{out} = d_6 = 64$, and $C_{in} = d_6 = 64$. We also have the kernel height and weight can be $kH = 2, kW = 2, sH = 2, sW = 2, dH = 1, dW = 1, padH = 0, padW = 0$, it means $kernelsize = 2$, $stride = 2$, and $padding = 0$, and $dilation = 1$, and $groups = 1$. And $C^{(L_8,Mp_0)} = 64$ and $H^{(L_8,Mp_0)} = W^{(L_8,Mp_0)} = 32$.

Refactor We than refactor ϕ into a new size such that $\phi^{(Ref_1)} \in \mathbb{R}^{S^{(Ref_1)}\times C^{(L_7,C_3)}/8\times(H^{(L_7,C_3)}\times W^{(L_7,C_3)}/4)}$, where $S^{(Ref_1)} = \frac{batch\times C^{(L_8,Mp_0)}\times H^{(L_8,Mp_0)}\times W^{(L_8,Mp_0)}}{C_{in}/8\times iH\times iW/4}$, for example, using the above $\phi^{(Ref_1)}$, we get new $\phi \in \mathbb{R}^{1\times64\times1024}$.

Attention Map - At A attention map is between θ and ϕ, we denote and $n = H^{(L_7,C_3)} \times W^{(L_7,C_3)}$, $m = C^{(L_7,C_3)}/8$, $p = H^{(L_7,C_3)} \times W^{(L_7,C_3)}/4$, we first reorder $\theta \in \mathbb{R}^{1\times n\times m}$, we then perform matrix-matrix product of $\theta \in \mathbb{R}^{b\times n\times m}$ and $\phi \in \mathbb{R}^{b\times m\times p}$. So the attention map is defined as

$$At := \theta \times_m \phi \to att \in \mathbb{R}^{b\times n\times p} \tag{35}$$

Here we have output $att \in \mathbb{R}^{1\times n\times p}$, where $n = 4096$ and $p = 1024$

Softmax A softmax layer is applied to the last dimension of att.

$$Softmax(att) :\to att^{(sfx)} \tag{36}$$

where

$$Softmax(a_i) : \frac{exp(a_i)}{\sum_j exp(a_j)} \tag{37}$$

Convolution - g **with max pooling** Let filter A as the weight matrix after spectral normalization, which is a tensor in $\mathbb{R}^{C_{out}\times\frac{C_{in}}{groups}\times kH\times kW}$, . And the input is in $\mathbb{R}^{batch\times C_{in}\times iH\times iW}$.

The convolution $\mathcal{C}_g$ is mapping from input $x \in \mathbb{R}^{N\times C^{(L_7,C_3)}\times H^{(L_7,C_3)}\times W^{(L_7,C_3)}}$ to $g \in \mathbb{R}^{N\times C_{out}\times oH\times oW}$ defined as:

$$\mathcal{C}_g(x) \to g \in \mathbb{R}^{1\times C_{out}\times oH\times oW} \tag{38}$$

We let $C_{out} = d_6 = 64$, and $C_{in} = d_9 = 512$. here we let $A \in \mathbb{R}^{256\times512\times1\times1}$ We let height and weight be: $iH = 64, iW = 64, C_{in} = 512$, the kernel height and weight can be $kH = 1, kW = 1, sH = 1, sW = 1, dH = 1, dW = 1, padH = 0, padW = 0$, it means $kernelsize = 1$, $stride = 1$, and $padding = 1$, and $dilation = 1$, and $groups = 1$. where $C^{C_g} = 256, H^{C_g} = 64, W^{C_g} = 64$.

A max pooling layer is followed after the $\mathcal{C}_g(x)$. Here we have input $g \in \mathbb{R}^{batch\times C\times iH\times iW}$. A max pooling layer is function mapping from input $\theta \in \mathbb{R}^{N\times C\times iH\times iW}$ to $x \in \mathbb{R}^{N\times C\times oH\times oW}$ defined as:

$$Maxpooling_g(g) \to g^{(MP_1)} \in \mathbb{R}^{1\times C\times oH\times oW} \tag{39}$$

We also have the kernel height and weight can be $kH = 2, kW = 2, sH = 2, sW = 2, dH = 1, dW = 1, padH = 0, padW = 0$, it means $kernelsize = 2$, $stride = 2$, and $padding = 0$, and $dilation = 1$, and $groups = 1$. where $C = 256, H^{(MP_1)} = 32, W^{(MP_1)} = 32$

And specifically,

$$H^{(MP_1)} = \frac{iH + 2\times padH - dH\times(kH-1) - 1}{sH} + 1$$
$$W^{(MP_1)} = \frac{iW + 2\times padW - dW\times(kW-1) - 1}{sW} + 1$$

Refactor We than refactor g into $g^{(Ref_3)} \in \mathbb{R}^{n^{(Ref_3)}\times C_{in}/2\times iH\times iW/4}$, and

$$nS = \frac{batch\times C_{out}\times oH\times oW}{C_{in}/2\times iH\times iW/4}$$

,

For the above g, we get $g^{(Ref_3)} \in \mathbb{R}^{1\times256\times1024}$.

Attention Map - g A attention map Att_g is between g and $att^{(sfx)}$, we denote $n = H^{(L_7,C_3)}\times W^{(L_7,C_3)}$, $m = C^{(L_7,C_3)}/8$, $p = H^{(L_7,C_3)}\times W^{(L_7,C_3)}/4$, we first reorder $\theta \in \mathbb{R}^{1\times n\times m}$, we then perform matrix-matrix product of $g \in \mathbb{R}^{b\times n\times m}$ and $att \in \mathbb{R}^{b\times m\times p}$. So the attention map is defined

$$\mathcal{A}_g : g \times_m att \to x^{(att_g)} \in \mathbb{R}^{b\times n\times p} \tag{40}$$

Here we have $att \in \mathbb{R}^{1\times1024\times4096}$, we have output $att_g^{(att_1)} \in \mathbb{R}^{1\times256\times4096}$

Refactor We than refactor $x^{(att_g)}$ into $x^{(att_g)} \in \mathbb{R}^{n^{ref_3}\times C^{(L_7,C_3)}/2\times H^{(L_7,C_3)}\times W^{(L_7,C_3)}}$, where $n^{ref_3} = \frac{b\times n\times p}{C^{(L_7,C_3)}/2\times H^{(L_7,C_3)}\times W^{(L_7,C_3)}}$.

For example, using the above $x^{(att_g)}$, we get new $x^{(att_g)} \in \mathbb{R}^{1\times256\times64\times64}$.

Convolution - o We let filter A as the weight matrix after spectral normalization, which is a tensor in $\mathbb{R}^{C_{out}\times\frac{C_{in}}{groups}\times kH\times kW}$. And the input is a tensor in $\mathbb{R}^{batch\times C_{in}\times iH\times iW}$.

The convolution $\mathcal{C}_o$ is mapping from input $att_g \in \mathbb{R}^{N\times C_{in}\times iH\times iW}$ to $att_o \in \mathbb{R}^{N\times C_{out}\times oH\times oW}$ defined as:

$$\mathcal{C}_o(att_g) \to x^{(C_o)} \in \mathbb{R}^{N\times C_{out}\times oH\times oW} \tag{41}$$

For example, in our case, we let $A \in \mathbb{R}^{512\times256\times1\times1}$. And we have $iH = 64, iW = 64, C_{in} = 256$. Other parameters are $kH = 1, kW = 1, sH = 1, sW = 1, dH = 1, dW = 1, padH = 0, padW = 0$, it means $kernelsize = 1$, $stride = 1$, and $padding = 1$, and $dilation = 1$, and $groups = 1$. where $N = 1, C^{(C_o)} = 512, H^{(C_o)} = 64, W^{(C_o)} = 64$

Addition The last layer is addition operation same as previous

$$x^{(L_7,C_3)} + \gamma \times x^{(C_o)} \to x^{(L_8,A_0)} \in \mathbb{R}^{N\times C^{(C_o)}\times H^{(C_o)}\times W^{(C_o)}} \tag{42}$$

where $N = 1, C^{(C_o)} = 512, H^{(C_o)} = 64, W^{(C_o)} = 64, \gamma = 1.992$.

3.3.5 Summary

For notation wise, we use g, gu, gud, s to denote the above four different modules. And the next module's input channel size is the previous module's output channel size.

So the x and c pass through series modules, we give the details in the following:

$$\mathcal{L}_0^g(x \in \mathbb{R}^{1\times2048\times4\times4}, c \in \mathbb{R}^{1\times256}, \sigma) :\to x \in \mathbb{R}^{1\times2048\times4\times4} \tag{43}$$

$$\mathcal{L}_1^{gu}(x, c, \sigma) :\to x \in \mathbb{R}^{1\times2048\times8\times8} \tag{44}$$

$$\mathcal{L}_2^{g}(x, c, \sigma) :\to x \in \mathbb{R}^{1\times2048\times8\times8} \tag{45}$$

$$\mathcal{L}_3^{gup}(x, c, \sigma) :\to x \in \mathbb{R}^{1\times1024\times16\times16} \tag{46}$$

$$\mathcal{L}_4^{g}(x, c, \sigma) :\to x \in \mathbb{R}^{1\times1024\times16\times16} \tag{47}$$

$$\mathcal{L}_5^{gu}(x, c, \sigma) :\to x \in \mathbb{R}^{1\times1024\times32\times32} \tag{48}$$

$$\mathcal{L}_6^{g}(x, c, \sigma) :\to x \in \mathbb{R}^{1\times1024\times32\times32} \tag{49}$$

$$\mathcal{L}_7^{gup}(x, c, \sigma) :\to x \in \mathbb{R}^{1\times512\times64\times64} \tag{50}$$

$$\mathcal{L}_8^{s}(x) :\to x \in \mathbb{R}^{1\times512\times64\times64} \tag{51}$$

$$\mathcal{L}_9^{g}(x, c, \sigma) :\to x \in \mathbb{R}^{1\times512\times64\times64} \tag{52}$$

$$\mathcal{L}_{10}^{gup}(x, c, \sigma) :\rightarrow x \in \mathbb{R}^{1\times 256\times 128\times 128} \tag{53}$$

$$\mathcal{L}_{11}^{g}(x, c, \sigma) :\rightarrow x \in \mathbb{R}^{1\times 256\times 128\times 128} \tag{54}$$

$$\mathcal{L}_{12}^{gup}(x, c, \sigma) :\rightarrow x \in \mathbb{R}^{1\times 128\times 256\times 256} \tag{55}$$

So in the end, we have sequential layers operator $\mathcal{L}$ defined as

$$\mathcal{L} : \mathcal{L}_{12}^{gup} \circ \cdots \circ \mathcal{L}_{0}^{g}(x_o, c, \sigma) \tag{56}$$

3.4 Batch Normalization with RelU

The last batch normalization is applied to the x after sequential layers. Here we define some pre-computed statistics for this operation. And the input channel size is $C^{(L_{12},C_3)}$. Let $f^{(B_0)}$, $s^{(B_0)}$ are the fractional and integer parts of $\frac{\sigma}{\eta}$. Let $r_{means}^{(B_0)} \in 0^{r_1^{(B_0)}\times C^{(L_{12},C_3)}}$ and $r_{vars}^{(B_0)} \in 1^{r_1^{(B_0)}\times C^{(L_{12},C_3)}}$, and $\eta = \frac{1}{r_1^{(B_0)} - r_2^{(B_0)}}$. And $r_{mean}^{(B_0)} = (r_{means}^{(B_0)})_{s^{(B_0)},:}$, $r_{var}^{(B_0)} = (r_{vars}^{(B_0)})_{s^{(B_0)},:}$ are sliced vectors from the matrix by setting the row index equals to $s^{(B_0)}$. Then we refactor $r_{mean}^{(B_0)} \in 0^{1\times C^{(L_{12},C_3)}\times 1\times 1}$, $r_{var}^{(B_0)} \in 1^{1\times C^{(L_{12},C_3)}\times 1\times 1}$.

Unlike the $\mathcal{B}_0$ which is a conditional batch normalization considering the c, The $\mathcal{B}$ create weight w and bias b by random initialization.

$$w \in 0^{1\times C^{(L_{12},C_3)}} \tag{57}$$

and we refactor $w \in \mathbb{R}^{1\times C^{(L_{12},C_3)}\times 1\times 1}$ and

$$b \in 0^{1\times C^{(L_{12},C_3)}} \tag{58}$$

and we refactor $b \in \mathbb{R}^{1\times C^{(L_{12},C_3)}\times 1\times 1}$.

And after training, we have the fixed value for w, b. So a batch normalization $\mathcal{B}$ is defined as

$$\mathcal{B} : \frac{x - r_{mean}}{\sqrt{r_{var} + \epsilon}} * w + b \rightarrow x \in \mathbb{R}^{1\times C^{(L_{12},C_3)}\times H^{(L_{12},C_3)}\times W^{(L_{12},C_3)}} \tag{59}$$

where $\epsilon = 0.0001$

Relu The rectified liner unit function is applied afterwards such that

$$ReLU(x) : max(0, x) \rightarrow x^{(R_o)} \in \mathcal{R}^{1\times 128\times 256\times 256} \tag{60}$$

We set hyperparameter $r_1^{(B_0)} = 51, r_2^{(B_0)} = 1$, where $C^{(L_{12},C_3)} = 128$, $H^{(L_{12},C_3)} = W^{(L_{12},C_3)} = 256$

3.5 Convolution with RGB

Convolution Let filter A as the weight matrix after spectral normalization, which is a tensor in $\mathbb{R}^{C^{Co} \times \frac{C_{in}}{groups}, kH, kW}$. And bias $b \in \mathbb{R}^{C_{out}}$, And the input is in $\mathbb{R}^{batch \times C_{in}, iH, iW}$.

The convolution with RGB $\mathbb{C}$ is mapping from input $x \in \mathbb{R}^{N \times C_{in} \times iH \times iW}$ to $x^{(C_o)} \in \mathbb{R}^{N \times C^{(Co)} \times H^{(Co)} \times W^{(Co)}}$ defined as:

$$\mathbb{C}(x) \to x^{(C_o)} \in \mathbb{R}^{1 \times 128 \times 256 \times 256} \tag{61}$$

Here we let $A \in \mathbb{R}^{128 \times 128 \times 3 \times 3}$ we let $b \in \mathbb{R}^{128}$. height and weight can be: $iH = 64, iW = 64, C_{in} = 128$, the kernel height and weight can be $kH = 1, kW = 1, sH = 1, sW = 1, dH = 1, dW = 1, padH = 1, padW = 1$, it means $kernelsize = 1$, $stride = 1$, and $padding = 1$, and $dilation = 1$, and $groups = 1$. We have $C^{(C_o)} = 128$, $H^{(C_o)} = W^{(C_o)} = 256$.

Slicing For the 128 channels in $x^{(C_o)}$, we select the first 3 channel from it. The slicing operator $\mathcal{S}$ is:

$$\mathcal{S} :\to x^{(S)} \mathbb{R}^{1 \times C^{(S)} \times H^{(S)} \times W^{(S)}} \tag{62}$$

where $C^{(S)} = 3$, and $H^{(S)} = W^{(S)} = 256$

3.6 Tanh activation

The last operation is the hyperbolic tangent tanh activation $tanh$ such that

$$tanh :\to x^{(Th)} \in \mathbb{R}^{1 \times C^{(S)} \times H^{(S)} \times W^{(S)}} \tag{63}$$

specifically, we have

$$\text{Tanh}(x) = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \tag{64}$$

So the output value is between $[-1, 1]$

4 Convert to Images

This section we convert the $x^{(Th)}$ into actual images.

4.1 Refactor

The refactor $Rcal_o$ is a operation that reorders x such that

$$Rcal :\to x^{(R_o)} \in \mathbb{R}^{1 \times H^{(S)} \times W^{(S)} \times C^{(S)}} \tag{65}$$

Then the original value is converted to $[0, 255]$ by $x = (x^{(R_o)} + 1)/2 \times 256)$. And we control the range of x falls into $[0, 255]$

5 Convolution layers Resnet-50

5.1 Transformer

The transformer re-scale, and normalize the original image. The original image is in RGB format that has dimension $\mathbb{R}^{H^{(S)} \times W^{(S)} \times C^{(S)}}$.

Resize Resize transformer Rs is defined such that:

$$Rs :\rightarrow x^{(Rs)} \in \mathbb{R}^{3\times H^{(Rs)}\times W^{(Rs)}} \tag{66}$$

where $H^{(Rs)} = W^{(Rs)} = 224$

Normalization A normalization $\mathcal{N}$ is a operator that applies to $x^{(Rs)}$ in each channel such that:

$$\mathcal{N} : \frac{x - mean(x)}{std(x)} \tag{67}$$

where $mean(x), std(x) \in \mathbb{R}^{C^{(S)}\times 1\times 1}$ Then we refactor $x^{(N)} \in \mathbb{R}^{1\times C^{(S)}\times H^{(Rs)}\times W^{(Rs)}}$

A embedding $e \in 0^{1\times C\times 1\times 1}$ will be used. where $C = 2048$

5.2 Convolution, batch normalization, relu, max-pooling layers

Convolution - $\mathcal{C}_0^{Res-50}$ Let filter A as the weight matrix, which is a tensor in $\mathbb{R}^{C^{(C_0^R)}\times \frac{C_{in}}{groups}, kH, kW}$, And the input is in $\mathbb{R}^{batch\times C_{in}, iH, iW}$,

The convolution $\mathcal{C}$ is mapping from input $x \in \mathbb{R}^{N\times C_{in}\times iH\times iW}$ to $x \in \mathbb{R}^{N\times C^{(C_0^{Res-50})}\times H^{(C_0^{Res-50})}\times W^{(C_0^{Res-50})}}$ defined as:

$$\mathcal{C}(x) \rightarrow x^{((C_0^{Res-50}))} \in \mathbb{R}^{1\times 64\times 112\times 112} \tag{68}$$

here we let $A \in \mathbb{R}^{64\times 3\times 7\times 7}$. height and weight can be: $iH = 254, iW = 254, C_{in} = 3$, the kernel height and weight can be $kH = 7, kW = 7, sH = 2, sW = 2, dH = 1, dW = 1, padH = 3, padW = 3$, it means $kernelsize = 7$, $stride = 2$, and $padding = 3$, and $dilation = 1$, and $groups = 1$., and $C_0^{Res-50} = 64$, $H_0^{Res-50} = W_0^{Res-50} = 112$

Batch Normalization The last batch normalization is applied to the x. The $\mathcal{B}$ use pre-computed statistics. $r_{means} \in 0^{C_0^{Res-50}}$ and $r_{vars} \in 1^{C_0^{Res-50}}$, and $momentum = 0.1$. After training,r_{means}, r_{vars} are returned from training data. the So $r_{mean} = momentum * input_{means} + (1 - momentum) * r_{means}$, $r_{var} = momentum * input_{vars} + (1 - momentum) * r_{vars}$. Then we refactor $r_{mean} \in 0^{1\times C_0^{Res-50}\times 1\times 1}$, $r_{var} \in 1^{1\times C_0^{Res-50}\times 1\times 1}$.

The input channel size is 64. And $\epsilon = 0.00001$

The $\mathcal{B}$ create weight w and bias b by random initialization.

$$w \in 0^{1\times C^{(C_0^{Res-50})}} \tag{69}$$

and we refactor $w \in \mathbb{R}^{1\times C^{(C_0^{Res-50})}\times 1\times 1}$ and

$$b \in 0^{1\times C^{(C_0^{Res-50})}} \tag{70}$$

and we refactor $b \in \mathbb{R}^{1\times C^{(C_0^{Res-50})}\times 1\times 1}$.

And after training, we have the fixed value for w, b. So a batch normalization $\mathcal{B}$ is defined as

$$\mathcal{B} : \frac{x - r_{mean}}{\sqrt{r_{var} + \epsilon}} * w + b \rightarrow x \in \mathbb{R}^{1\times C^{(C_0^{Res-50})}\times H^{(C_0^{Res-50})}\times W^{(C_0^{Res-50})}} \tag{71}$$

Relu The rectified liner unit function is applied afterwards such that

$$ReLU(x) : max(0, x) \rightarrow x^{R_0^{Res-50}} \in \mathcal{R}^{1\times 64\times 112\times 112} \tag{72}$$

Max Pooling Layer A max pooling layer is followed after the $\mathcal{C}_\phi(\phi)$, Here we have input $\phi \in \mathbb{R}^{batch,C,iH,iW}$. We also have the kernel height and weight can be $kH = 3, kW = 3, sH = 2, sW = 2, dH = 1, dW = 1, padH = 1, padW = 1$, it means $kernelsize = 3$, $stride = 2$, and $padding = 1$, and $dilation = 1$, and $groups = 1$. A max pooling layer is function mapping from input $\theta \in \mathbb{R}^{N\times C\times iH\times iW}$ to $x^{(MP_0^{Res-50})} \in \mathbb{R}^{N\times C\times oH\times oW}$ defined as:

$$Maxpooling_g(g) \rightarrow x^{(MP_0^{Res-50})} \in \mathbb{R}^{1\times 64\times 56\times 56} \tag{73}$$

where

$$H = \frac{iH + 2 \times padH - dH \times (kH - 1) - 1}{sH} + 1 \tag{74}$$

$$W = \frac{iW + 2 \times padW - dW \times (kW - 1) - 1}{sW} + 1 \tag{75}$$

where $C^{(MP_0^{Res-50})} = 64$, and $H^{(MP_0^{Res-50})} = W^{(MP_0^{Res-50})} = 56$

5.3 Sequential Layers

One sequential layers $\mathcal{S}$ could contain the following layers:

A bottleneck with down-sample layers, a convolution layer and a batch normalization layer and a down-sample layer, then a relu layer. For simplicity, we give this kind of bottleneck $\mathcal{L}^d$

A bottleneck without down-sample layers, a convolution layer and a batch normalization layer, then a relu layer. We give this kind of bottleneck as $\mathcal{L}^b$

Convolution layer Let filter A as the weight matrix, which is a tensor in $\mathbb{R}^{C_{out}\times \frac{C_{in}}{groups},kH,kW}$, And the input could be the output of max pooling layer, or the previous sequential layer. So we use $x^{(S)} \in \mathbb{R}^{1\times C^{(S)}\times H^{(S)}\times W^{(S)}}$ to implicitly represent the input.

The convolution $\mathcal{C}_1^{Res-50}$ is mapping from input $x^{(S)} \in \mathbb{R}^{N\times C_{in}\times iH\times iW}$ to $x \in \mathbb{R}^{N\times C_{out}\times oH\times oW}$ defined as:

$$\mathcal{C}(x) \rightarrow x \in \mathbb{R}^{1\times C^{(C_1^{Res-50})}\times H^{(C_1^{Res-50})}\times W^{(C_1^{Res-50})}} \tag{76}$$

Suppose the input is the $x^{(MP_0^{Res-50})}$ here we let $A \in \mathbb{R}^{64\times 64\times 1\times 1}$, we let height and weight can be: $iH = 56, iW = 56, C_{in} = 64$, the kernel height and weight can be $kH = 1, kW = 1, sH = 1, sW = 1, dH = 1, dW = 1, padH = 0, padW = 0$, it means $kernelsize = 1$, $stride = 1$, and $padding = 0$, and $dilation = 1$, and $groups = 1$. We have $C^{(C_1^{Res-50})} = 64$ and $H^{(C_1^{Res-50})} = W^{(C_1^{Res-50})} = 56$

Batch Normalization layer - $\mathcal{B}_1^{S,Res-50}$ The last batch normalization is applied to the x. The $\mathcal{B}_1^{Res-50}$ use pre-computed statistics. The input channel size is $C^{(C_1^{Res-50})}$. $r_{means} \in 0^{C^{(C_1^{Res-50})}}$ and $r_{vars} \in 1^{C^{(C_1^{Res-50})}}$, and $momentum =$

0.1. After training,r_{means}, r_{vars} are returned from training data. the So $r_{mean} = momentum * input_{means} + (1 - momentum) * r_{means}$, $r_{var} = momentum * input_{vars} + (1 - momentum) * r_{vars}$. Then we refactor $r_{mean} \in 0^{1\times(C_1^{Res-50})\times 1\times 1}$, $r_{var} \in 1^{1\times C^{(C_1^{Res-50})}\times 1\times 1}$. And $\epsilon = 0.00001$

The $\mathcal{B}$ create weight w and bias b by random initialization.

$$w \in 0^{1\times C^{(C_1^{Res-50})}} \tag{77}$$

and we refactor $w \in \mathbb{R}^{1\times C^{(C_1^{Res-50})}\times 1\times 1}$ and

$$b \in 0^{1\times C^{(C_1^{Res-50})}} \tag{78}$$

and we refactor $b \in \mathbb{R}^{1\times C^{(C_1^{Res-50})}\times 1\times 1}$.

And after training, we have the fixed value for w, b. So a batch normalization $\mathcal{B}$ is defined as

$$\mathcal{B} : \frac{x - r_{mean}}{\sqrt{r_{var} + \epsilon}} * w + b \to x^{(B_1^{(Res-50)})} \in \mathbb{R}^{1\times C^{(C_1^{Res-50})}\times H^{(C_1^{Res-50})}\times W^{(C_1^{Res-50})}} \tag{79}$$

Relu layer The rectified liner unit function is applied afterwards such that

$$ReLU(x) : max(0, x) \to x^{R_1^{Res-50}} \in \mathbb{R}^{1\times 64\times 56\times 56} \tag{80}$$

Convolution layer - $\mathcal{C}_2^{(Res-50)}$ Let filter A as the weight matrix, which is a tensor in $\mathbb{R}^{C_{out}\times \frac{C_{in}}{groups}, kH, kW}$, And the input is in $\mathbb{R}^{batch\times C_{in}, iH, iW}$.

The convolution $\mathcal{C}$ is mapping from input $x \in \mathbb{R}^{N\times C_{in}\times iH\times iW}$ to $x \in \mathbb{R}^{N\times C_{out}\times oH\times oW}$ defined as:

$$\mathcal{C}(x) \to x^{(C_2^{Res-50})} \in \mathbb{R}^{1\times 64\times 56\times 56} \tag{81}$$

Here we let $A \in \mathbb{R}^{64\times 64\times 3\times 3}$. The input height and weight can be: $iH = 56, iW = 56, C_{in} = 64$, the kernel height and weight can be $kH = 3, kW = 3, sH = 1, sW = 1, dH = 1, dW = 1, padH = 1, padW = 1$, it means $kernelsize = 3$, $stride = 1$, and $padding = 1$, and $dilation = 1$, and $groups = 1$. We have $C^{(C_2^{Res-50})} = 64$, and $H^{(C_2^{Res-50})} = W^{(C_2^{Res-50})} = 56$

Batch Normalization layer - $\mathcal{B}_2^{Res-50}$ Same operation as previous batch normalization layer, here a batch normalization $\mathcal{B}_2^{Res-50}$ is defined as

$$\mathcal{B} : \frac{x - r_{mean}}{\sqrt{r_{var} + \epsilon}} * w + b \to x \in \mathbb{R}^{1\times 64\times 56\times 56} \tag{82}$$

Relu layer - $\mathcal{R}_2^{(Res-50)}$ The rectified liner unit function is applied afterwards such that

$$ReLU(x) : max(0, x) \to x^{(R_2^{(Res-50)})} \in \mathbb{R}^{1\times C^{(C_2^{Res-50})}\times H^{(C_2^{Res-50})}\times W^{(C_2^{Res-50})}} \tag{83}$$

Convolution layer - $\mathcal{C}_3^{Res-50}$ Let filter A be the weight matrix, which a tensor in $\mathbb{R}^{C^{(C_3^{Ret-50})} \times \frac{C_{in}}{groups}, kH, kW}$,

The convolution $\mathcal{C}$ is mapping from input $x \in \mathbb{R}^{N \times C_{in} \times iH \times iW}$ to $x \in \mathbb{R}^{N \times C_{out} \times oH \times oW}$ defined as:

$$\mathcal{C}(x) \to x^{(C_3^{Ret-50})} \in \mathbb{R}^{1 \times C^{(C_3^{Ret-50})} \times H^{(C_3^{Ret-50})} \times W^{(C_3^{Ret-50})}} \tag{84}$$

here we let $A \in \mathbb{R}^{256 \times 64 \times 1 \times 1}$. And the input is in $\mathbb{R}^{batch \times C_{in}, iH, iW}$ height and weight can be: $iH = 56, iW = 56, C_{in} = 64$, the kernel height and weight can be $kH = 1, kW = 1, sH = 1, sW = 1, dH = 1, dW = 1, padH = 0, padW = 0$, it means $kernelsize = 1$, $stride = 1$, and $padding = 1$, and $dilation = 1$, and $groups = 1$. We have $C^{(C_3^{Ret-50})} = 256$, $H^{(C_3^{Ret-50})} = W^{(C_3^{Ret-50})} = 56$

Batch Normalization layer Same operation as previous batch normalization layer, here a batch normalization $\mathcal{B}$ is defined as

$$\mathcal{B} : \frac{x - r_{mean}}{\sqrt{r_{var} + \epsilon}} * w + b \to x^{(B_3)} \in \mathbb{R}^{1 \times 256 \times 56 \times 56} \tag{85}$$

Down Sampling Down sampling composes of multiple layers, including: another convolution layer, batch normalization layer, the input is the original input x_o. So we use the convolution layer to define it.

Let filter A as the weight matrix, which is a tensor in $\mathbb{R}^{C_{out} \times \frac{C_{in}}{groups}, kH, kW}$, And the input is in $\mathbb{R}^{batch \times C_{in}, iH, iW}$,

The convolution $\mathcal{C}d$ is mapping from input $x^{(MP_0^{Res-50})} \in \mathbb{R}^{N \times C_{in} \times iH \times iW}$ to $x^d \in \mathbb{R}^{N \times C_{out} \times oH \times oW}$ defined as:

$$\mathcal{C}d(x^{(MP_0^{Res-50})}) \to x^d \in \mathbb{R}^{1 \times C^d \times H^d \times W^d} \tag{86}$$

here we let $A \in \mathbb{R}^{256 \times 64 \times 1 \times 1}$. height and weight can be: $iH = 56, iW = 56, C_{in} = 64$, the kernel height and weight can be $kH = 1, kW = 1, sH = 1, sW = 1, dH = 1, dW = 1, padH = 0, padW = 0$, it means $kernelsize = 1$, $stride = 1$, and $padding = 1$, and $dilation = 1$, and $groups = 1$. $iC^{C_1^{Res-50}} = 64$ and $iW^{C_1^{Res-50}} = iH^{C_1^{Res-50}} = 56$. And we have $C^d = 256$, $H^d = W^d = 56$

Same operation as previous batch normalization layer, here a batch normalization $\mathcal{B}$ is defined as

$$\mathcal{B}d : \frac{x^d - r_{mean}^{bd}}{\sqrt{r_{var}^{bd} + \epsilon}} * w + b \to x^{bd} \in \mathbb{R}^{1 \times C^d \times H^d \times W^d} \tag{87}$$

Addition The layer before the last layer is an addition operation

$$x^{(B_3)} + x^{bd} \to x^{(A^{Res-50})} \in \mathbb{R}^{1 \times C^{(C_3^{Ret-50})} \times H^{(C_3^{Ret-50})} \times W^{(C_3^{Ret-50})}} \tag{88}$$

Relu layer The rectified liner unit function is applied afterwards such that

$$ReLU(x) : max(0, x) \to x^{(R_3^{Res-50})} \in \mathcal{R}^{1 \times C^{(C_3^{Ret-50})} \times H^{(C_3^{Ret-50})} \times W^{(C_3^{Ret-50})}} \tag{89}$$

where $C^{(C_3^{Ret-50})} = 256$, $H^{(C_3^{Ret-50})} = 56$ $W^{(C_3^{Ret-50})} = 56$.

Summary of Sequential Layers Here we give a summary of the sequential layers.

The next bottleneck module's input channel size is the previous module's output channel size.

So the $x^{(MP_0^{Res-50})}$ pass through series modules, we give the details in the following:

$\mathcal{S}_1$ sequential layer 1:

$$\mathcal{L}_1^d(x^{(MP_0^{Res-50})} \in \mathbb{R}^{1\times 64\times 56\times 56}) :\rightarrow x^{S_1,L_1^d} \in \mathbb{R}^{1\times C^{((S_1,L_1^d))}\times H^{(S_1,L_1^d)}\times W^{(S_1,L_1^d)}} \tag{90}$$

where $C^{((S_1,L_1^d))} = 256$, and $H^{(S_1,L_1^d)} = H^{(S_1,L_1^d)} = 56$

In detail,

$$\begin{aligned}\mathcal{L}_1^d(x^{(MP_0^{Res-50})}) :&\mathcal{C}(x^{(MP_0^{Res-50})} \in \mathbb{R}^{1\times 64\times 56\times 56}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times 64\times 56\times 56}) \rightarrow Relu(x \in \mathbb{R}^{1\times 64\times 56\times 56}) \rightarrow \\ &\mathcal{C}(x \in \mathbb{R}^{1\times 64\times 56\times 56}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times 64\times 56\times 56}) \rightarrow Relu(x \in \mathbb{R}^{1\times 64\times 56\times 56}) \rightarrow \\ &\mathcal{C}(x \in \mathbb{R}^{1\times 64\times 56\times 56}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times 256\times 56\times 56}) \rightarrow Downsampling(x^{(MP_0^{Res-50})} \in \mathbb{R}^{1\times 64\times 56\times 56}) \\ &Addition(x \in \mathbb{R}^{1\times 256\times 56\times 56}, x_o \in \mathbb{R}^{1\times 256\times 56\times 56}) \rightarrow Relu(x \in \mathbb{R}^{1\times 256\times 56\times 56})\end{aligned} \tag{91}$$

And,

$$\mathcal{L}_2^b(x) :\rightarrow x^{S_1,L_2^b} \in \mathbb{R}^{1\times 256\times 56\times 56} \tag{92}$$

In detail,

$$\begin{aligned}\mathcal{L}_2^b(x_o) :&\mathcal{C}_1^{(S_1,L_2^b,Res-50)}(x_o \in \mathbb{R}^{1\times 256\times 56\times 56}) \rightarrow \mathcal{B}_1^{(S_1,L_2^b,Res-50)}(x \in \mathbb{R}^{1\times 64\times 56\times 56}) \rightarrow Relu_1^{(S_1,L_2^b,Res-50)}(x \in \mathbb{R}^{1\times \\ &\mathcal{C}_2^{(S_1,L_2^b,Res-50)}(x \in \mathbb{R}^{1\times 64\times 56\times 56}) \rightarrow \mathcal{B}_2^{(S_1,L_2^b,Res-50)}(x \in \mathbb{R}^{1\times 64\times 56\times 56}) \rightarrow Relu_2^{(S_1,L_2^b,Res-50)}(x \in \mathbb{R}^{1\times 64 \\ &\mathcal{C}_3^{(S_1,L_2^b,Res-50)}(x \in \mathbb{R}^{1\times 256\times 56\times 56}) \rightarrow \mathcal{B}_3^{(S_1,L_2^b,Res-50)}(x \in \mathbb{R}^{1\times 64\times 56\times 56}) \rightarrow \\ &Addition_1^{(S_1,L_2^b,Res-50)}(x \in \mathbb{R}^{1\times 256\times 56\times 56}, x_o \in \mathbb{R}^{1\times 256\times 56\times 56}) \rightarrow Relu_3^{(S_1,L_2^b,Res-50)}(x \in \mathbb{R}^{1\times 256\times 56\times 56})\end{aligned} \tag{93}$$

$$\mathcal{L}_3^b(x) :\rightarrow x^{S_1,L_3^b} \in \mathbb{R}^{1\times 256\times 56\times 56} \tag{94}$$

In detail,

$$\begin{aligned}\mathcal{L}_3^b(x_o) :&\mathcal{C}(x_o \in \mathbb{R}^{1\times 256\times 56\times 56}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times 64\times 56\times 56}) \rightarrow Relu(x \in \mathbb{R}^{1\times 64\times 56\times 56}) \rightarrow \\ &\mathcal{C}(x \in \mathbb{R}^{1\times 64\times 56\times 56}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times 64\times 56\times 56}) \rightarrow Relu(x \in \mathbb{R}^{1\times 64\times 56\times 56}) \rightarrow \\ &\mathcal{C}(x \in \mathbb{R}^{1\times 64\times 56\times 56}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times 256\times 56\times 56}) \rightarrow \\ &Addition(x \in \mathbb{R}^{1\times 256\times 56\times 56}, x_o \in \mathbb{R}^{1\times 256\times 56\times 56}) \rightarrow Relu(x \in \mathbb{R}^{1\times 256\times 56\times 56})\end{aligned} \tag{95}$$

$\mathcal{S}_2$ sequential layer 2:

$$\mathcal{L}_1^d(x_o \in \mathbb{R}^{1\times256\times56\times56}) :\rightarrow x^{(S_2,L_1^d)} \in \mathbb{R}^{1\times256\times56\times56} \tag{96}$$

In detail,

$$\begin{aligned}\mathcal{L}_1^d(x_o) :&\mathcal{C}(x_o \in \mathbb{R}^{1\times256\times56\times56}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times128\times56\times56}) \rightarrow Relu(x \in \mathbb{R}^{1\times128\times56\times56}) \rightarrow \\ &\mathcal{C}(x \in \mathbb{R}^{1\times128\times56\times56}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times128\times28\times28}) \rightarrow Relu(x \in \mathbb{R}^{1\times128\times28\times28}) \rightarrow \\ &\mathcal{C}(x \in \mathbb{R}^{1\times128\times28\times28}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times512\times28\times28}) \rightarrow Downsampling(x_o^{1\times256\times56\times56}) \rightarrow \\ &Addition(x_d \in \mathbb{R}^{1\times512\times28\times28}, x \in \mathbb{R}^{1\times512\times28\times28}) \rightarrow Relu(x \in \mathbb{R}^{1\times512\times28\times28})\end{aligned} \tag{97}$$

And,

$$\mathcal{L}_2^b(x) :\rightarrow x^{(S_2,L_2^b)} \in \mathbb{R}^{1\times512\times28\times28} \tag{98}$$

In detail,

$$\begin{aligned}\mathcal{L}_2^b(x_o) :&\mathcal{C}(x_o \in \mathbb{R}^{1\times512\times28\times28}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times128\times28\times28}) \rightarrow Relu(x \in \mathbb{R}^{1\times128\times28\times28}) \rightarrow \\ &\mathcal{C}(x \in \mathbb{R}^{1\times128\times28\times28}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times128\times28\times28}) \rightarrow Relu(x \in \mathbb{R}^{1\times128\times28\times28}) \rightarrow \\ &\mathcal{C}(x \in \mathbb{R}^{1\times128\times28\times28}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times512\times28\times28}) \rightarrow \\ &Addition(x \in \mathbb{R}^{1\times512\times28\times28}, x_o \in \mathbb{R}^{1\times512\times28\times28}) \rightarrow Relu(x \in \mathbb{R}^{1\times512\times28\times28})\end{aligned} \tag{99}$$

$$\begin{aligned}\mathcal{L}_3^b(x) :=& \mathcal{C}_1^{(S_2,L_3^b,Res-50)} \circ \mathcal{B}_1^{(S_2,L_3^b,Res-50)} \circ Relu_1^{(S_2,L_3^b,Res-50)} && (100)\\ &\circ \mathcal{C}_2^{(S_2,L_3^b,Res-50)} \circ \mathcal{B}_2^{(S_2,L_3^b,Res-50)} \circ Relu_2^{(S_2,L_3^b,Res-50)} && (101)\\ &\circ \mathcal{C}_3^{(S_2,L_3^b,Res-50)} \circ \mathcal{B}_3^{(S_2,L_3^b,Res-50)} \circ Addition_1^{(S_2,L_3^b,Res-50)} \circ Relu_3^{(S_2,L_3^b,Res-50)}(x) && (102)\\ &\rightarrow x^{(S_2,L_3^b)} \in \mathbb{R}^{1\times512\times28\times28} && (103)\end{aligned}$$

$$\mathcal{L}_4^b(x) := \mathcal{C} \circ \mathcal{B} \circ Relu \circ \mathcal{C} \circ \mathcal{B} \circ Relu \circ \mathcal{C} \circ \mathcal{B} \circ Addition \circ Relu(x) \rightarrow x^{(S_2,L_4^b)} \in \mathbb{R}^{1\times512\times28\times28} \tag{104}$$

$\mathcal{S}_3$ sequential layer 3:

$$\mathcal{L}_1^d(x^{(S_2,L_4^b)} \in \mathbb{R}^{1\times512\times28\times28}) :\rightarrow x^{(S_3,L_1^d)} \in \mathbb{R}^{1\times256\times56\times56} \tag{105}$$

In detail,

$$\begin{aligned}\mathcal{L}_1^d(x_o) :&\mathcal{C}(x_o \in \mathbb{R}^{1\times512\times28\times28}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times256\times28\times28}) \rightarrow Relu(x \in \mathbb{R}^{1\times256\times28\times28}) \rightarrow \\ &\mathcal{C}(x \in \mathbb{R}^{1\times256\times28\times28}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times256\times14\times14}) \rightarrow Relu(x \in \mathbb{R}^{1\times256\times14\times14}) \rightarrow \\ &\mathcal{C}(x \in \mathbb{R}^{1\times256\times14\times14}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times1024\times14\times14}) \rightarrow Downsampling(x_o^{1\times512\times28\times28}) \rightarrow \\ &Addition(x_d \in \mathbb{R}^{1\times1024\times14\times14}, x \in \mathbb{R}^{1\times1024\times14\times14}) \rightarrow Relu(x \in \mathbb{R}^{1\times1024\times14\times14})\end{aligned} \tag{106}$$

And,

$$\mathcal{L}_2^b(x) :\rightarrow x^{(S_3,L_2^b)} \in \mathbb{R}^{1\times\times 28\times 28} \tag{107}$$

In detail,

$$\begin{aligned}\mathcal{L}_2^b(x_o) :&\mathcal{C}(x_o \in \mathbb{R}^{1\times 1024\times 14\times 14}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times 256\times 14\times 14}) \rightarrow Relu(x \in \mathbb{R}^{1\times 256\times 14\times 14}) \rightarrow \\ &\mathcal{C}(x \in \mathbb{R}^{1\times 256\times 14\times 14}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times 256\times 14\times 14}) \rightarrow Relu(x \in \mathbb{R}^{1\times 256\times 14\times 14}) \rightarrow \\ &\mathcal{C}(x \in \mathbb{R}^{1\times 256\times 14\times 14}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times 1024\times 14\times 14}) \rightarrow \\ &Addition(x \in \mathbb{R}^{1\times 1024\times 14\times 14}, x_o \in \mathbb{R}^{1\times 1024\times 14\times 14}) \rightarrow Relu(x \in \mathbb{R}^{1\times 1024\times 14\times 14})\end{aligned} \tag{108}$$

$$\mathcal{L}_3^b(x) := \mathcal{C}\circ\mathcal{B}\circ Relu\circ\mathcal{C}\circ\mathcal{B}\circ Relu\circ\mathcal{C}\circ\mathcal{B}\circ Addition\circ Relu(x) \rightarrow x^{(S_3,L_3^b)} \in \mathbb{R}^{1\times 1024\times 14\times 14} \tag{109}$$

$$\mathcal{L}_4^b(x) := \mathcal{C}\circ\mathcal{B}\circ Relu\circ\mathcal{C}\circ\mathcal{B}\circ Relu\circ\mathcal{C}\circ\mathcal{B}\circ Addition\circ Relu(x) \rightarrow x^{(S_3,L_4^b)} \in \mathbb{R}^{1\times 1024\times 14\times 14} \tag{110}$$

$$\mathcal{L}_5^b(x) := \mathcal{C}\circ\mathcal{B}\circ Relu\circ\mathcal{C}\circ\mathcal{B}\circ Relu\circ\mathcal{C}\circ\mathcal{B}\circ Addition\circ Relu(x) \rightarrow x^{(S_3,L_5^b)} \in \mathbb{R}^{1\times 1024\times 14\times 14} \tag{111}$$

$$\mathcal{L}_6^b(x) := \mathcal{C}\circ\mathcal{B}\circ Relu\circ\mathcal{C}\circ\mathcal{B}\circ Relu\circ\mathcal{C}\circ\mathcal{B}\circ Addition\circ Relu(x) \rightarrow x^{(S_3,L_6^b)} \in \mathbb{R}^{1\times 1024\times 14\times 14} \tag{112}$$

$\mathcal{S}_4$ sequential layer 4:

$$\mathcal{L}_1^d(x^{(S_3,L_6^b)} \in \mathbb{R}^{1\times 1024\times 14\times 14}) :\rightarrow x^{(S_4,L_1^d)} \in \mathbb{R}^{1\times 256\times 56\times 56} \tag{113}$$

In detail,

$$\begin{aligned}\mathcal{L}_1^d(x_o) :&\mathcal{C}(x_o \in \mathbb{R}^{1\times 1024\times 14\times 14}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times 512\times 14\times 14}) \rightarrow Relu(x \in \mathbb{R}^{1\times 512\times 14\times 14}) \rightarrow \\ &\mathcal{C}(x \in \mathbb{R}^{1\times 512\times 14\times 14}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times 512\times 7\times 7}) \rightarrow Relu(x \in \mathbb{R}^{1\times 512\times 7\times 7}) \rightarrow \\ &\mathcal{C}(x \in \mathbb{R}^{1\times 512\times 7\times 7}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times 2048\times 7\times 7}) \rightarrow Downsampling(x_o^{1\times 1024\times 14\times 14}) \rightarrow \\ &Addition(x_d \in \mathbb{R}^{1\times 2048\times 7\times 7}, x \in \mathbb{R}^{1\times 2048\times 7\times 7}) \rightarrow Relu(x \in \mathbb{R}^{1\times 2048\times 7\times 7})\end{aligned} \tag{114}$$

And,

$$\mathcal{L}_2^b(x) :\rightarrow x^{(S_4,L_2^b)} \in \mathbb{R}^{1\times 2048\times 7\times 7} \tag{115}$$

In detail,

$$\mathcal{L}_2^b(x_o) :\mathcal{C}(x_o \in \mathbb{R}^{1\times 2048\times 7\times 7}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times 512\times 7\times 7}) \rightarrow Relu(x \in \mathbb{R}^{1\times 512\times 7\times 7}) \rightarrow \tag{116}$$
$$\mathcal{C}(x \in \mathbb{R}^{1\times 512\times 7\times 7}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times 512\times 7\times 7}) \rightarrow Relu(x \in \mathbb{R}^{1\times 512\times 7\times 7}) \rightarrow$$
$$\mathcal{C}(x \in \mathbb{R}^{1\times 512\times 7\times 7}) \rightarrow \mathcal{B}(x \in \mathbb{R}^{1\times 2048\times 7\times 7}) \rightarrow$$
$$Addition(x \in \mathbb{R}^{1\times 2048\times 7\times 7}, x_o \in \mathbb{R}^{1\times 2048\times 7\times 7}) \rightarrow Relu(x \in \mathbb{R}^{1\times 2048\times 7\times 7})$$

$$\mathcal{L}_3^b(x) := \mathcal{C} \circ \mathcal{B} \circ Relu \circ \mathcal{C} \circ \mathcal{B} \circ Relu \circ \mathcal{C} \circ \mathcal{B} \circ Addition \circ Relu(x) \rightarrow x^{(S_4,L_3^b)} \in \mathbb{R}^{1\times C^{((S_4,L_3^b))}\times H^{(S_4,L_3^b)}\times H^{(S_4,L_3^b)}} \tag{117}$$

where $C^{((S_4,L_4^b))} = 2048$, and $H^{(S_4,L_4^b)} = W^{(S_4,L_4^b)} = 7$

$$\mathcal{L}_4^b(x) := \mathcal{C} \circ \mathcal{B} \circ Relu \circ \mathcal{C} \circ \mathcal{B} \circ Relu \circ \mathcal{C} \circ \mathcal{B} \circ Addition \circ Relu(x) \rightarrow x^{(S_4,L_4^b)} \in \mathbb{R}^{1\times C^{((S_4,L_4^b))}\times H^{(S_4,L_4^b)}\times H^{(S_4,L_4^b)}} \tag{118}$$

where $C^{((S_4,L_4^b))} = 2048$, and $H^{(S_4,L_4^b)} = W^{(S_4,L_4^b)} = 7$

5.4 Adaptive Average Pooling layer

An adaptive average pooling layer will calculate the kernel size based on the input and output size. Here we have input $x^{(S_4,L_2^b)} \in \mathbb{R}^{1\times C^{((S_4,L_4^b))}\times H^{(S_4,L_4^b)}\times W^{(S_4,L_4^b)}}$, and we have output $x^{(adp)} \in \mathbb{R}^{1\times C^{(adp)}\times H^{(adp)}\times W^{(adp)}}$. So we have input size as $ips = H^{(S_4,L_4^b)} \times W^{(S_4,L_4^b)}$, and we have output size as $ops = H^{(adp)} \times W^{(adp)}$. We also have the kernel size as $\lfloor \frac{ips+ops-1}{ops} \rfloor$.

An adaptive pooling layer is function mapping from input $x^{(S_4,L_2^b)} \in \mathbb{R}^{1\times C^{((S_4,L_4^b))}\times H^{(S_4,L_4^b)}\times W^{(S_4,L_4^b)}}$ to $x^{(adp)} \in \mathbb{R}^{1\times C^{(adp)}\times H^{(adp)}\times W^{(adp)}}$ defined as:

$$x^{(adp)} \in \mathbb{R}^{1\times C^{(adp)}\times H^{(adp)}\times W^{(adp)}} := AdaptiveAvgPooling(x^{(S_4,L_2^b)}) \tag{119}$$

where

$$H^{(adp)} = \frac{iH + 2 \times padH - dH \times (kH - 1) - 1}{sH} + 1 \tag{120}$$
$$W^{(adp)} = \frac{iW + 2 \times padW - dW \times (kW - 1) - 1}{sW} + 1 \tag{121}$$

Here, the $iH = iW = 7$, and the $oH = oW = 1$, So the kernel height and weight can be $kH = 49, kW = 49$, it means $kernelsize = 49$, we have $C^{(adp)} = 2048$, $H^{(adp)} = W^{(adp)} = 1$

Output Representation We are interested in this layer, so the output $x^{(rep)}$ will be retrieved from this layer. So the final $x^{(rep)} \in \mathbb{R}^{C^{(adp)}}$

For example, we get some good statistics of $x^{(rep)}$ such that $mean(x^{(rep)}) = 0.42$ and $std(x^{(rep)}) = 0.47$, $max(x^{(rep)}) = 5.04$ and $min(x^{(rep)}) = 0.0$

5.5 A fully Connected layer

We refactor $x^{(adp)} \in \mathbb{R}^{1\times C^{(adp)}\times 1\times 1}$ to $x^{(adp)} \in \mathbb{R}^{1\times C^{(adp)}}$. And a fully connected layer is after this,

A fully connected layer is a linear transformation layer such that

$$\mathcal{F} : \alpha x^{(adp)} A^T + \beta b \to x^{out} \in \mathbb{R}^{1\times |L|} \tag{122}$$

where $A \in \mathbb{R}^{|L|\times C^{(adp)}}$ is a weight matrix, and $b \in \mathbb{R}^{1\times |L|}$ is bias vector, and parameters $\alpha = 1, \beta = 1$

So the final output after resnet is $x^{out} \in \mathbb{R}^{1\times |L|}$

6 Simulation of GAN and GMM

6.1 Simulation Experiment

Given the model of GAN and CNN representation, we simulate our representation data from GAN, and constructed the corresponding GMM from the following process. Suppose we have K classes, and for each class, we let x_i is a vector with dimension p, where p depends on the model of the CNN, which represents the one of the layers of CNN. We have total image sample size N for each class. We let m of N images to construct the statistics of GMM, and the rest $n = N - m$ to do representation of GAN and GMM samples, where m is a parameter between $[1, N]$. We let $\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$, we construct the noise vector $\mathbf{t} = [N(0,1)\dots N(0,1)]$.

We let $j = 1\dots n$, $\{z_j, y_j\}$ represent the GMM sample and GAN sample respectively. Then we use the following calculations to construct our sample.

$$\underbrace{z_j}_{\text{dimension: } 1\times p} = \bar{x} + \underbrace{\mathbf{t}}_{\text{p dimension vector}} (\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T)^{\frac{1}{2}} \tag{123}$$

$$\underbrace{y_j}_{\text{dimension: } 1\times p} = x_i, i = 1\dots n \tag{124}$$

We use previous procedure generate the n samples, so the total sample size is Kn. We denote as $\mathbf{X}$. The Gram matrix $G \in \mathbb{R}^{Kn}$ was calculated as below.

$$G = \frac{1}{p}\mathbf{X}\mathbf{X}^T \tag{125}$$

Then eigenvalues of Gram matrix for both the images and GMM were calculated, which is a Kn dimension vector.

6.2 Experiment Settings

In our simulation experiment, we choose $K = 3$. We also choose $N = 500$, based on the CNN models we choose, our p could equal to 2048, 4096, or 1920, we choose $m = 330$, then $n = 500 - 330 = 170$,

Based on the previous result, $\mathbf{X}$ is a $510 \times p$ dimension matrix., then the eigenvalues will be the 510 dimension vector. We then conducted the following experiments.

6.3 Distribution of Noise Vector z

We change the sampling distribution of noise vector z. We let the z distributed from uniform $U(0,1)$, $Bernoulli(0,1)$ with $P = 0.5$, standard normal spherical Gaussian with mean 0 and covariance matrix $\sigma = I$, Gaussian with mean 0 and a covariance matrix with its entry $\sigma_{ij} = 0.5$, and Gaussian with mean 0 and a covariance matrix with its entry $\sigma_{ij} = 1.0$.

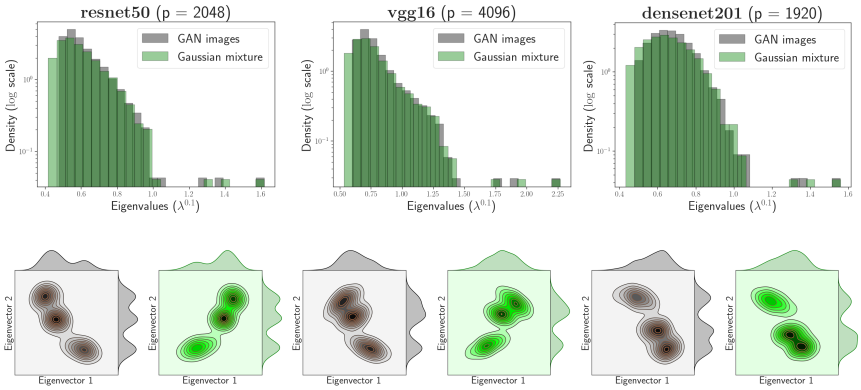


Figure 1: Histogram of The Gram Matrix of GAN Data Given $z \sim U(-1,1)$ and GMM with Same Mean and Covariance

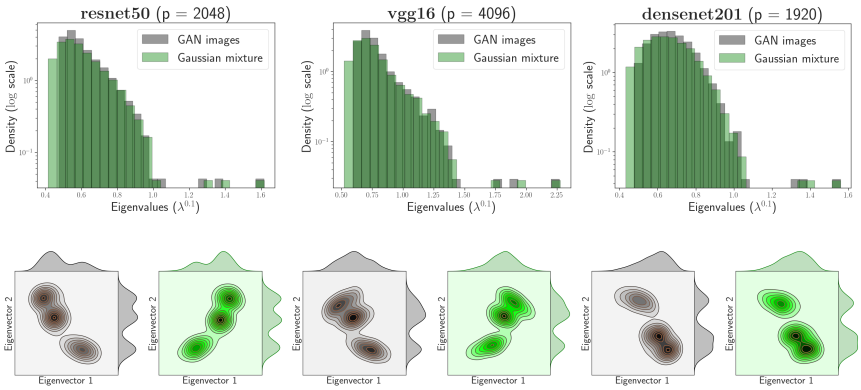


Figure 2: Histogram of The Gram Matrix of GAN Data Given $z \sim Bernoulli(0,1)$ and GMM with Same Mean and Covariance

Fig 1, Fig 2, Fig 3 show the distribution of eigenvalues of Gram matrix under different distributions of noise vector z, we have seen that the the eigenvalue of Gram matrix is invariant to distribution of input of the Generator.

6.4 Distribution of Noise Vector t

The GMM is constructed in equation 120, where the covariance is constructed to be the same with the Gram matrix of the GAN data. t in equation 120 is noise vector with

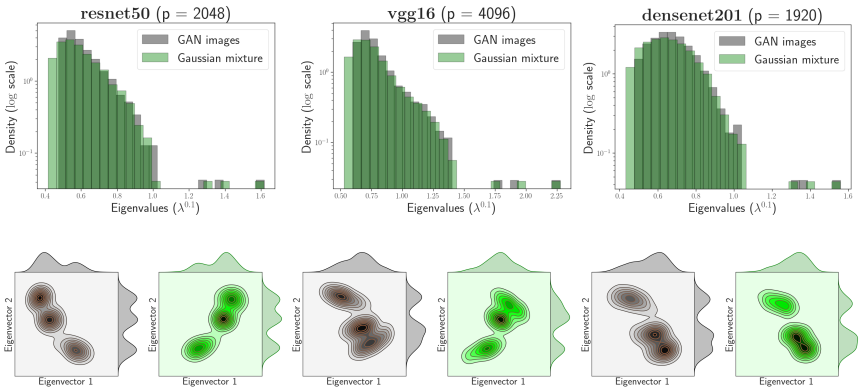


Figure 3: Histogram of The Gram Matrix of GAN Data Given $z \sim N(0,1)$ and GMM with Same Mean and Covariance

distribution of $N(0,1)$. We then investigated the spectral behavior of the data after we change the distribution of t,

We first fix the distribution of $z \sim N(0,1)$, then change the distribution of t such that

$$t_j = (t_{1j}, \ldots, t_{pj}) \in R^p \sim Beroulli(-1,1)$$

. Fig 1 shows the density of eigenvalues of the Gram matrix under such scenario.

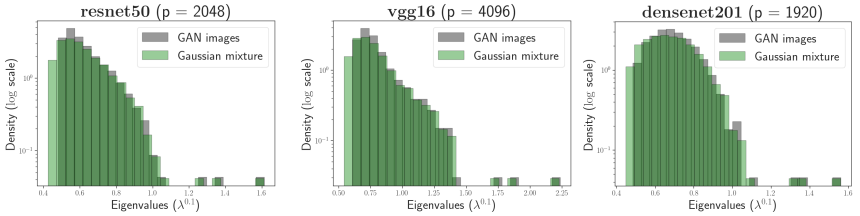


Figure 4: Histogram of The Gram Matrix of GAN Data Given $z \sim N(0,1)$ and Constructed Model Given $t \sim Bernoulli(-1,1)$

We then change the distribution of $z \sim Bernoulli(0,1)$, and let the t has distribution

$$t_j = (t_{1j}, \ldots, t_{pj}) \in R^p \sim Bernoulli(-1,1)$$

. Fig 5 shows the density of eigenvalues of the Gram matrix under this scenario.

Fig 4 and Fig 5 show the distribution of eigenvalues the Gram matrix still resumes the spectral behavior when the data follows a model rather than standard GMM, where the means remain the same but covariance are different.

7 MNIST Extension

We extends the result into a different GAN settings for MNIST data set.

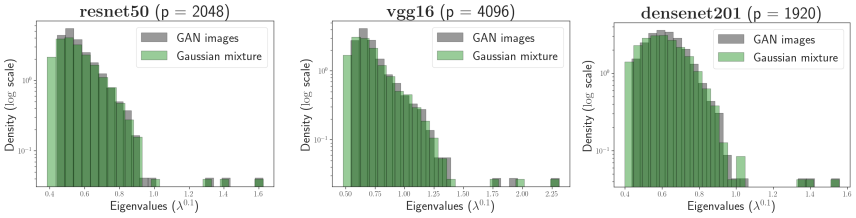


Figure 5: Histogram of The Gram Matrix of GAN Data Given $z \sim Bernoulli(0, 1)$ and Constructed Model Given $t \sim Bernoulli(-1, 1)$

7.1 Model of MNIST

We constructed a simple MNIST GAN network, where the discriminator has five fully connected hidden layers with leaky ReLU activation, and the generator has give fully connected layers with leaky ReLu activation followed with a batch normalization. The generated images then pass through a pre-trained CNN network for multi-classification. The CNN contains a two convolution 2D layers with a max pooling layer.

7.2 Generation of MNIST Images

We give the generator a noise vector $z \in R^{100} = N(0, 1)$, and train the neural network with 50 epochs. We sample the generated images during the training, and Fig 6 shows that the generator is able to generate better images over the training.

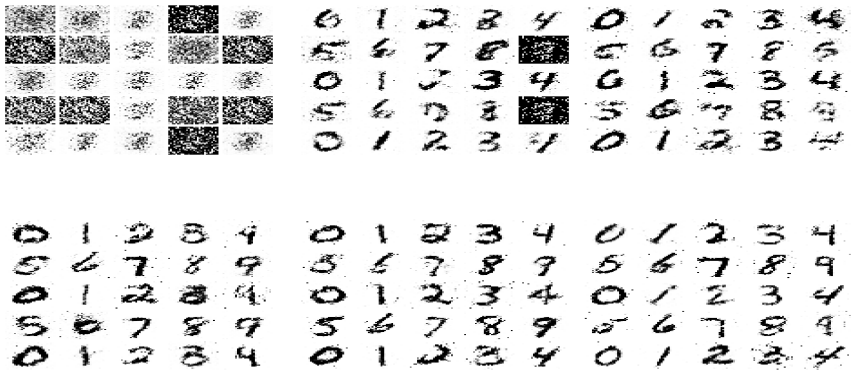


Figure 6: training samples

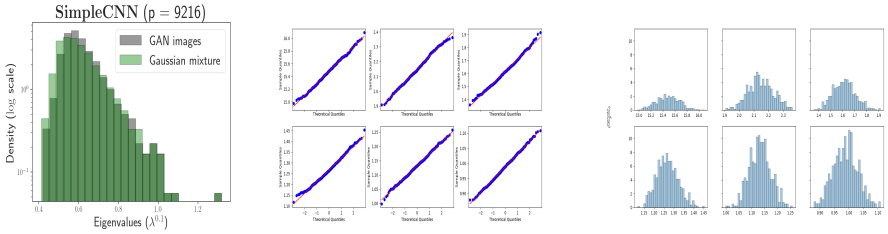
7.3 Simulation Experiment

In the simulation experiment, we have $k = 10$ classes. We generated $N = 200$ images per class, after passed through CNN, we get the layer output before the second last layer, where the $p = 9216$. We let $n = 50$, so the total matrix $X \in \mathbb{R}^{500\times 9216}$

Fig 7(a) gives the distribution of the Gram matrix of the above GAN data and corresponding GMM, we see that they have same distribution.

7.4 Simulation of Multiple Experiments

To futher invesitigate the distribution of the Gram matrix of the GAN, we run 500 simulations of the matrix and get the distribution of the second to sixth eigenvalues, and draw the QQ plot and the histogram of them.



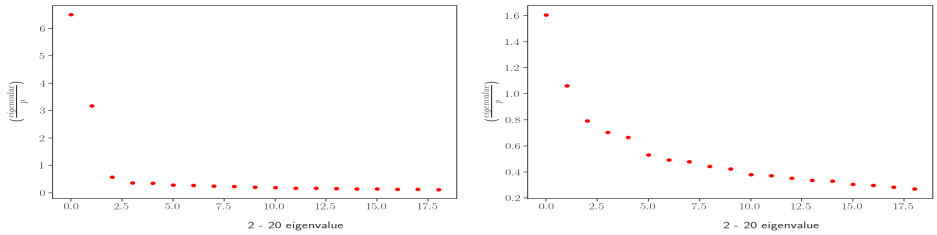
(a) GAN-Data and GMM-Data (b) QQ Plot (c) Histogram

Figure 7: (b) is the QQ plot of 500 simulations of the second to sixth eigenvalues, (c) is the histogram distribution.

Fig 7(b) and Fig 7(c) show the distribution of second to sixth eigenvalues of Gram matrix of the above GAN data over 500 simulations, we can see that they hold a normal distribution.

8 Change The Parameters of Generator

We change the parameters of the generator from 10, the original weights are trained based on the Imagenet. However, we give a similar eigenvalue structure of weights, by sampling the weights value from a distribution and keeping the relative maganitute of the weights, but not the exact value. Our claim is that the eigenvalue structure is invariant to the specific weights.



(a) Pre-Trained GAN (b) GAN with $\mathcal{C}_0$ as Untrained

Figure 8: (a) is the second to sixth eigenvalues of Gram matrix of GAN-data with pre-trained GAN, (b) is its counterpart with the convolution layer $\mathcal{C}_0$'s weights untrained and sampled from a distribution.

Fig 8 indicates that the eigenvalue behavior of the Gram matrix is invariant to the exact GAN weights, but is related to the structure of the weights.

References

[1] Brock, A., Donahue, J., and Simonyan, K. (2019). Large Scale GAN Training for High Fidelity Natural Image Synthesis. *arXiv:1809.11096 [cs, stat]*.

[2] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

[3] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Networks. *arXiv:1406.2661 [cs, stat]*.

[4] Seddik, M. E. A., Louart, C., Tamaazousti, M., and Couillet, R. (2020). Random Matrix Theory Proves that Deep Learning Representations of GAN-data Behave as Gaussian Mixtures. *arXiv:2001.08370 [cs, stat]*.