Problem 2.1

$S = w_0 + w_1 x_1 + w_2 x_2$

$S_1 = 4 + 2(-1) + 1(-1) = 1$

$S_2 = 4 + 2(-1) + 1(+1) = 3$

$S_3 = 4 + 2(+1) + 1(-1) = 5$

$S_4 = 4 + 2(1) + 1(1) = 7$

x_1	x_2	S	y
-1	-1	1	1
-1	+1	3	1
+1	-1	5	1
+1	+1	7	1

The Logic Function that this Adaline is perfoming is a Tautology

Problem 2.2

$w^{k+1} = w^k + r \, sgn(d_k) \, x_k$

Assume

$w_0 = 2.1$, $w_1 = 1$, $w_2 = 3$, $r = 2$

$S_1 = 2.1 + (1)(-1) + (3)(-1) = -1.9$

x_1	x_2	S	y
-1	-1		+1
-1	+1		+1
+1	-1		+1
+1	+1		-1

$w_1(new) = 1 + 2(1)(-1) = -1$

$w_2(new) = 3 + 2(1)(-1) = 1$

$S_1 = 2.1 + (-1)(-1) + (1)(-1)$

$= 2.1$

$S_2 = 2.1 + (-1)(-1) + (1)(1) = 4.1$

$S_3 = 2.1 + (-1)(1) + (1)(-1) = 2.1$

$S_4 = 2.1 + (-1)(1) + (1)(1) = 2.1$

$w_1 (new) = -1 + 2(-1)(1) = -3$

$w_2 (new) = 1 + 2(-1)(+1) = -1$

Calculating s with the new weights.

$S_1 = 2.1 + (-1)(-3) + (-1)(-1) = 7.1$

$S_2 = 2.1 + (-1)(-3) + (-1)(1) = 4.1$

$S_3 = 2.1 + (1)(-3) + (-1)(-1) = 0.1$

$S_4 = 2.1 + (1)(-3) + (1)(-1) = -1.9$.

x_1	x_2	S	y
-1	-1	7.1	+1
-1	+1	4.1	+1
+1	-1	0.1	+1
+1	+1	-1.9	-1

Problem 2.3

To implement a majority vote functionality. I'm configuring the model with a bias of zero and setting all the weights that is w_1, w_2 and w_3 equal to each other.

Assume $w_0 = 0$, $w_1 = 2$, $w_2 = 2$, $w_3 = 2$.

$S = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$

x_1	x_2	x_3	S	y
-1	-1	-1	-6	-1
-1	-1	+1	-2	-1
-1	+1	-1	-2	-1
-1	+1	+1	+2	+1
+1	-1	-1	-2	-1
+1	-1	+1	+2	+1
+1	+1	-1	+2	+1
+1	+1	+1	+6	+1

Problem 2.4

1^{st} Adaline

$w_0 = 0.5, \ w_1 = -1, \ w_2 = 1$

$S_1 = 0.5 + (-1)(-1) + (1)(-1) = 0.5$

$S_2 = 0.5 + (-1)(-1) + (1)(1) = 2.5$

$S_3 = 0.5 + (-1)(1) + (1)(-1) = -1.5$

$S_4 = 0.5 + (-1)(1) + (1)(1) = 0.5$

x_1	x_2	s	y
-1	-1	0.5	+1
-1	+1	2.5	+1
+1	-1	-1.5	-1
+1	+1	0.5	+1

2^{nd} Adaline.

$w_0 = 0.5, \ w_1 = 1, \ w_2 = -1$

$S_1 = 0.5 + (-1)(1) + (-1)(-1) = 0.5$

$S_2 = 0.5 + (-1)(1) + (-1)(1) = -1.5$

$S_3 = 0.5 + (1)(1) + (-1)(-1) = 2.5$

$S_4 = 0.5 + (1)(1) + (-1)(1) = 0.5$

x_1	x_2	s	y
-1	-1	0.5	+1
-1	+1	-1.5	-1
+1	-1	2.5	+1
+1	+1	0.5	+1

Assume.

$w_0 = 2, \ w_1 = -1, \ w_2 = -2.$

$S_1 = 2 + (1)(-1) + (1)(-2) = -1$

$S_2 = 2 + (1)(-1) + (-1)(-2) = 3$

$S_3 = 2 + (-1)(-1) + (-2)(1) = 1$

$S_4 = 2 + (-1)(1) + (-2)(1) = -1$

x_1	x_2	s	y
+1	+1	-1	-1
+1	-1	3	+1
-1	+1	1	+1
+1	+1	-1	-1

Problem 3.1

$$L = \frac{1}{2} (t - x_3)^T (t - x_3)$$

$$x_2 = \text{sigmoid}(w_1 x_1 + b_1)$$

$$x_3 = w_2 x_2 + b_2$$

(i) $$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial x_3} \times \frac{\partial x_3}{\partial x_2} \times \frac{\partial x_2}{\partial w_1}$$

$$\cancel{\frac{\partial L}{\partial x_3}} = \text{Let } (t - x_3) = p.$$

$$\frac{\partial L}{\partial x_3} = \frac{\partial L}{\partial p} \times \frac{\partial p}{\partial x_3} \Rightarrow \frac{\partial L}{\partial p} = \frac{\partial}{\partial p}\left[\frac{1}{2} p^T p\right]$$

$$= \frac{1}{2} \times 2p = p.$$

$$\frac{\partial L}{\partial x_3} = p. \frac{\partial}{\partial x_3}[t - x_3] = (t - x_3)(-1)$$

$$= x_3 - t.$$

$$\cancel{\frac{\partial L}{\partial w_1}} = \frac{\partial x_3}{\partial x_2} = w_2.$$

$$\frac{\partial x_2}{\partial w_1} = \frac{x_1 e^{x_1 w_1 + b_1}}{(1 + e^{x_1 w_1 + b_1})^2}$$

$$\therefore \frac{\partial L}{\partial w_1} = (x_3 - t)(w_2)\left[\frac{x_1 e^{x_1 w_1 + b_1}}{(1 + e^{x_1 w_1 + b_1})^2}\right]$$

ii) $\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial x_3} \times \frac{\partial x_3}{\partial w_2}$

$= (x_3 - t)(x_2)$

iii) $\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial x_3} \times \frac{\partial x_3}{\partial x_2} \times \frac{\partial x_2}{\partial b_1}$

$= (x_3 - t)(w_2) \times \left[\frac{e^{x_1 w_1 + b_1}}{(1 + e^{x_1 w_1 + b_1})^2}\right]$

iv) $\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial x_3} \times \frac{\partial x_3}{\partial b_2}$

$= (x_3 - t)$.

Problem 3.2.

$x_2 = \mathrm{Relu}(w_1 x_1 + b_1)$

$= \max(0, w_1 x_1 + b_1)$.

$$\frac{\partial L}{\partial w_1} = \begin{cases} (x_3 - t)(w_2)(x_1) & ; \; w_1 x_1 + b_1 > 0. \\ 0 & ; \; \text{otherwise} \end{cases}$$

$\frac{\partial L}{\partial w_2} = (x_3 - t)(x_2)$.

$$\frac{\partial L}{\partial b_1} = \begin{cases} (x_3 - t)(w_2). & ; w_1 x_1 + b_1 > 0. \\ 0 & ; \text{otherwise} \end{cases}$$

$$\frac{\partial L}{\partial b_2} = (x_3 - t).$$

Calculate x_2.

$$w_1 x_1 + b_1 = \begin{bmatrix} 3 & -1 & 1 \\ -5 & 2 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

Calculate x_3.

$$w_2 x_2 + b_2 = \begin{bmatrix} 1 & -2 \\ -3 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} -2 \\ -4 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -3 \end{bmatrix}$$

$$\frac{\partial L}{\partial w_1} = (x_3 - t)(w_2)(x_1).$$

$$= \left(\begin{bmatrix} -1 \\ -3 \end{bmatrix} - \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right) \begin{bmatrix} 1 & -2 \\ -3 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 13 & 26 \\ 0 & -1 & -2 \end{bmatrix}$$

$$\frac{\partial L}{\partial \omega_2} = (x_3 - t)\, x_2$$

$$= \underset{2\times1}{\begin{bmatrix} -2 \\ -5 \end{bmatrix}} \underset{1\times2}{\begin{bmatrix} 2 & 2 \end{bmatrix}}.$$

$$= \begin{bmatrix} -4 & -4 \\ -10 & -10 \end{bmatrix}$$

$$\frac{\partial L}{\partial b_1} = (x_3 - t)\, \omega_2$$

$$= \underset{2\times1}{\begin{bmatrix} -2 \\ -5 \end{bmatrix}} \underset{2\times2}{\begin{bmatrix} 1 & -2 \\ -3 & 1 \end{bmatrix}}.$$

$$= \begin{bmatrix} -2 & -5 \end{bmatrix} \begin{bmatrix} 1 & -2 \\ -3 & 1 \end{bmatrix}.$$

$$= \begin{bmatrix} 13 & -1 \end{bmatrix}$$

$$\frac{\partial L}{\partial b_2} = (x_3 - t) = \begin{bmatrix} -2 \\ -5 \end{bmatrix}.$$

ECE 661 Homework-1

Problem 1:

True/False Questions

1. **False**. To enhance a neural network's ability to generalize, one can either augment the training dataset by increasing the number of samples or reduce the model's complexity by decreasing the number of parameters. Increasing the number of parameters can make the model overfit the noise.
2. **False.** The final set of weights got after training depends on the initialization values.
3. **True.** Using a full batch size provides a more accurate gradient estimate, reducing the likelihood of becoming trapped in local minima during training.
4. **False.** An if else statement can split the model into two parts where each part is differentiable, for example a condition which says n>0, the backpropagation training algorithm cannot be applied to the entire model, only part of the model depending on the condition.
5. **False.** If the padding size and stride size are both set to one, enlarging the height and width of the kernel size will result in a smaller output feature map size.

Problem 4:

2D convolution:

Input matrix:

```
([[ 0.,  0., -1.,  0.,  0.,  0.,  1.,  0.,  0.],
  [ 0., -1., -1., -1.,  0.,  1.,  1.,  1.,  0.],
  [-1., -1., -1., -1.,  0.,  1.,  1.,  1.,  1.],
  [ 0., -1., -1., -1.,  0.,  1.,  1.,  1.,  0.],
  [ 0.,  0., -1.,  0.,  0.,  0.,  1.,  0.,  0.]])
```

Kernel:

```
([[ 0.0000, -0.5000,  0.0000],
  [-0.5000,  1.0000, -0.5000],
  [ 0.0000, -0.5000,  0.0000]])
```

Output matrix:

```
[[ 0.   1.  -0.5  1.   0.  -1.   0.5 -1.   0. ]
 [ 1.   0.   1.   0.   0.   0.  -1.   0.  -1. ]
 [-0.5  1.   1.   0.5  0.  -0.5 -1.  -1.   0.5]
 [ 1.   0.   1.   0.   0.   0.  -1.   0.  -1. ]
 [ 0.   1.  -0.5  1.   0.  -1.   0.5 -1.   0. ]]
```

Applying Kernel to an image

Original Image

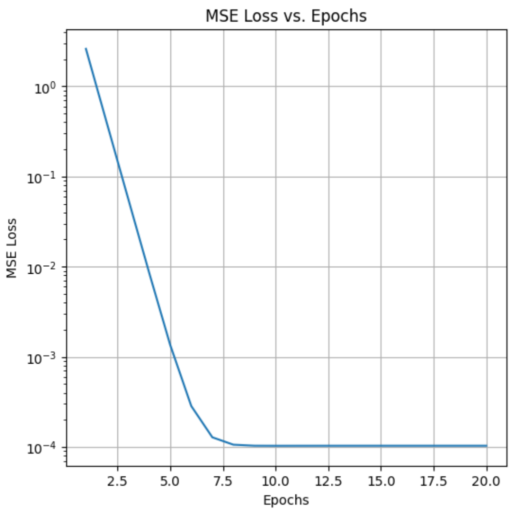
Feature Map After Convolution

When the kernel is applied to the image of the dog, results in inversion. This happens because the kernel contains both positive and negative values, and during the convolution operation, these values interact with the pixel values of the image. Specifically, the positive central value amplifies the importance of the central pixel, while the negative values subtract from the surrounding pixels. This combination of operations results in the image appearing inverted or flipped.

Lab 1: LMS Algorithm

Problem 5(d):

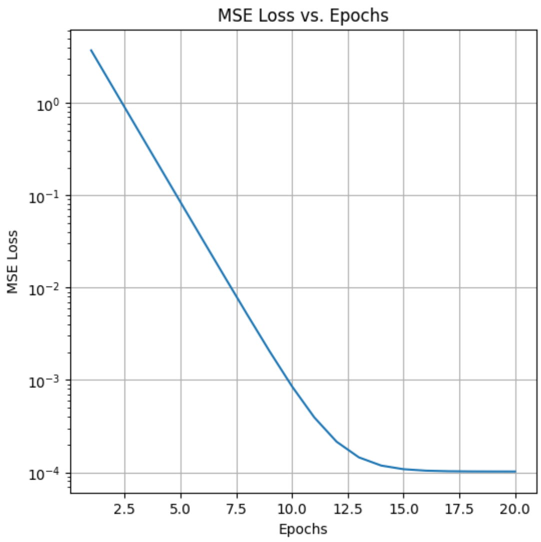
MSE vs Epochs when learning rate=0.01



Optimal weight when learning rate=0.01

```
[[ 1.00074855]
 [ 1.00082859]
 [-2.00068123]]
```

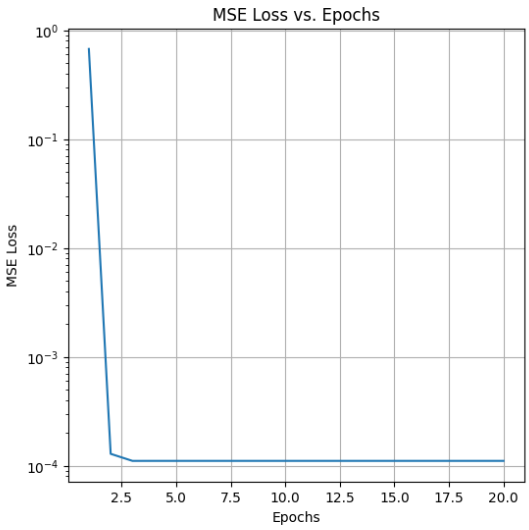
MSE vs Epochs when learning rate=0.005



Optimal weight when learning rate=0.005

```
[[ 1.00068274]
 [ 1.0006024 ]
 [-2.00033003]]
```

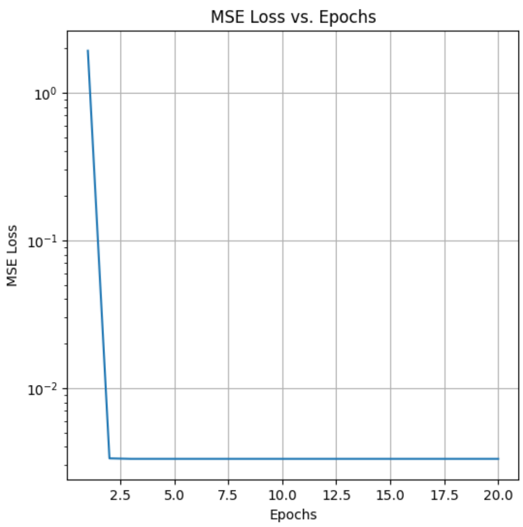
MSE vs Epochs when learning rate=0.05



Optimal weight when learning rate=0.05

```
[[ 1.00053087]
 [ 1.00163155]
 [-2.00162854]]
```

MSE vs Epochs when learning rate=0.5



Optimal weight when learning rate=0.5

```
[[ 0.97969496]
 [ 0.98520802]
 [-1.9666911 ]]
```

As the learning rate increases, convergence occurs more rapidly, but it must be carefully chosen, as an excessively high learning rate can cause the model to overshoot the minimum.

Lab 2: Simple NN

Problem 6(a)

```
# Create the neural network module: LeNet-5
class SimpleNN(nn.Module):
    def __init__(self):
        super(SimpleNN, self).__init__()
        # Layer definition
        self.conv1 =     CONV(in_channels=3, out_channels=16, kernel_size=5, stride=1, padding=2)#Your code here
        self.conv2 =     CONV(in_channels=16, out_channels=16, kernel_size=3, stride=1, padding=2)#Your code here
        self.conv3 =     CONV(in_channels=16, out_channels=32, kernel_size=7, stride=1, padding=2)#Your code here
        self.fc1   =     FC(in_features=288, out_features=32)#Your code here
        self.fc2   =     FC(in_features=32, out_features=10)#Your code here
```

```
def forward(self, x):
    # Forward pass computation
    # Conv 1
    out = F.relu(self.conv1(x))
    # MaxPool
    out = F.max_pool2d(out, 4, 2)
    # Conv 2
    out = F.relu(self.conv2(out))
    # MaxPool
    out = F.max_pool2d(out, 3, 2)
    # Conv 3
    out = F.relu(self.conv3(out))
    # MaxPool
    out = F.max_pool2d(out, 2, 2)
    # Flatten
    out = out.view(out.size(0), -1)
    # FC 1
    out = F.relu(self.fc1(out))
    # FC 2
    out= F.relu(self.fc2(out))
    return out
```

```
device = 'cuda' if torch.cuda.is_available() else 'cpu'
if device =='cuda':
    print("Run on GPU...")
else:
    print("Run on CPU...")

# Model Definition
net = SimpleNN()
net = net.to(device)

# Test forward pass
data = torch.randn(5,3,32,32)
data = data.to(device)
# Forward pass "data" through "net" to get output "out"
out =   net(data)   #Your code here

# Check output shape
assert(out.detach().cpu().numpy().shape == (5,10))
print("Forward pass successful")
```

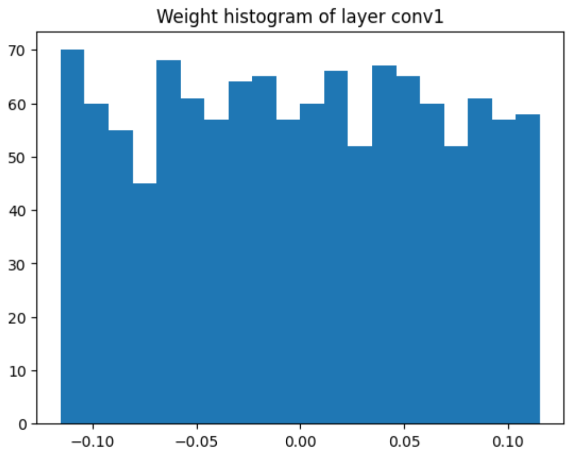
```
Run on CPU...
Forward pass successful
```

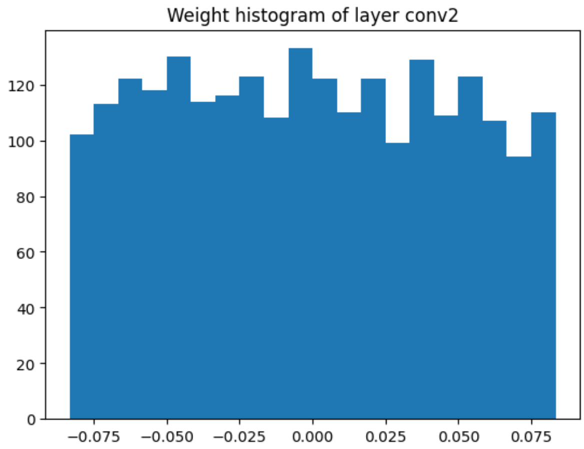
Problem 6(b)

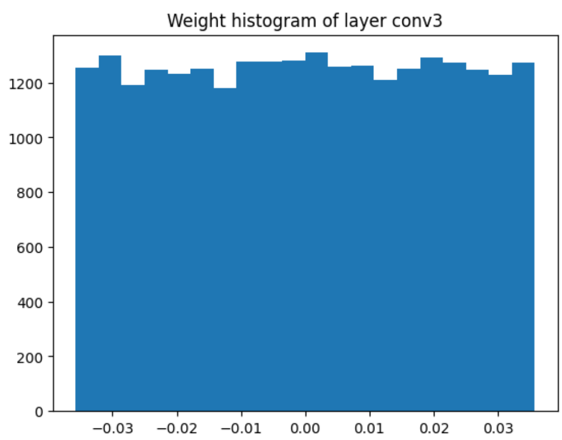
Layer	Input Shape	Output shape	Weight shape	# Param	#MAC
Conv1	(1, 3, 32, 32)	(1, 16, 32, 32)	(16, 3, 5, 5)	1200	1228800
Conv2	(1, 16, 15, 15)	(1, 16, 17, 17)	(16, 16, 3, 3)	2304	665856
Conv3	(1, 16, 8, 8)	(1, 32, 6, 6)	(32, 16, 7, 7)	25088	903168
FC1	(1, 288)	(1, 32)	(32, 288)	9216	9216
FC2	(1, 32)	(1, 10)	(10,32)	330	330

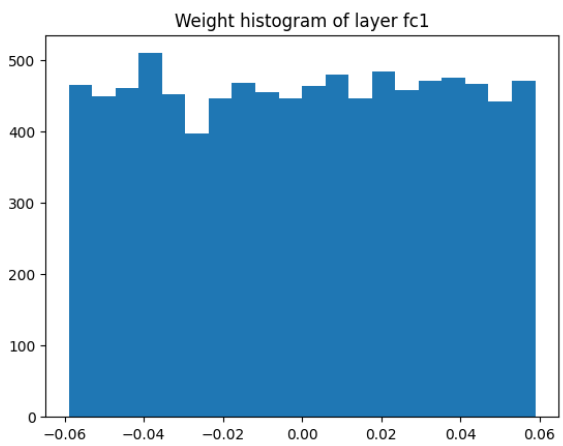
Lab 3: Bonus Question

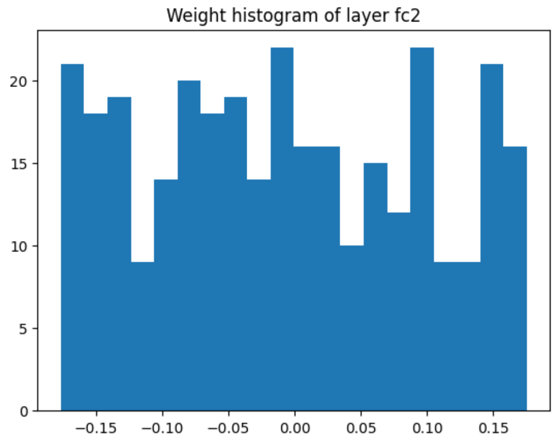
Weight Histogram:



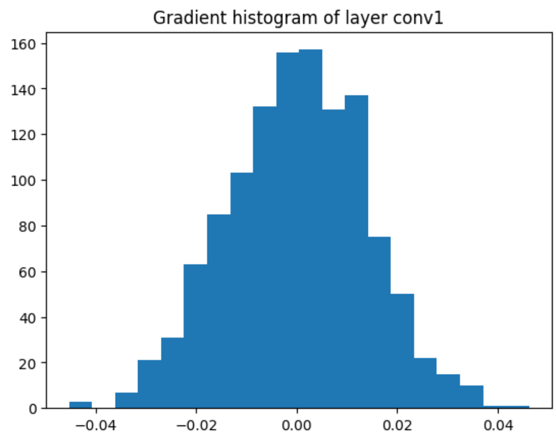


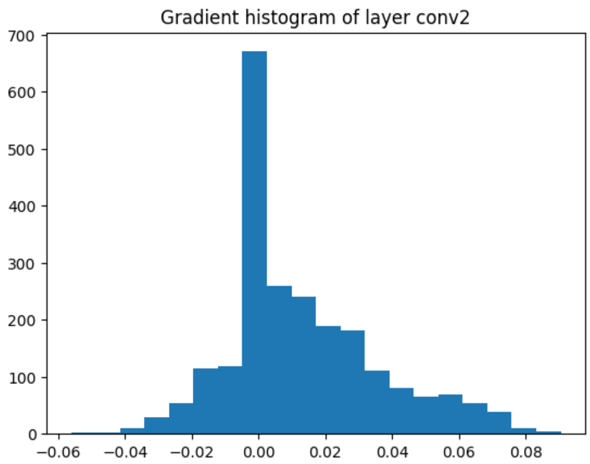


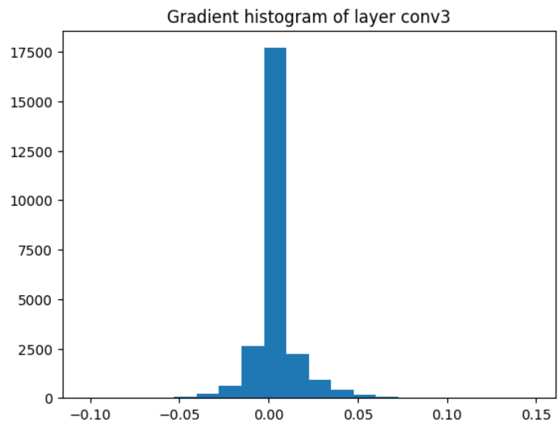


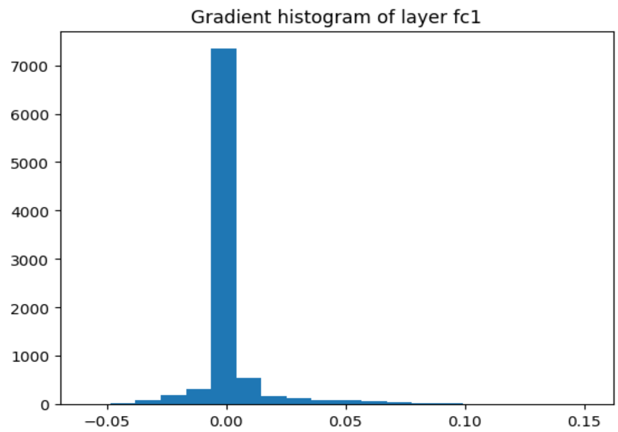


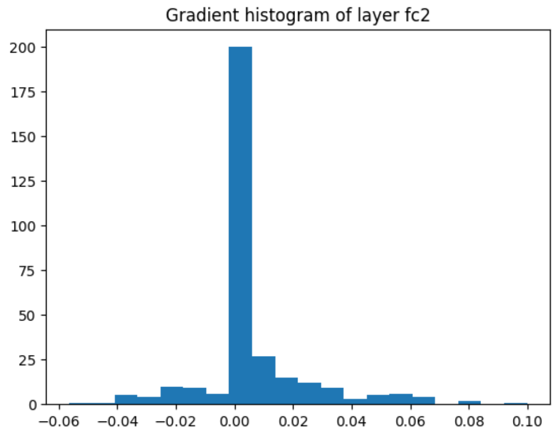
Gradient Histogram:



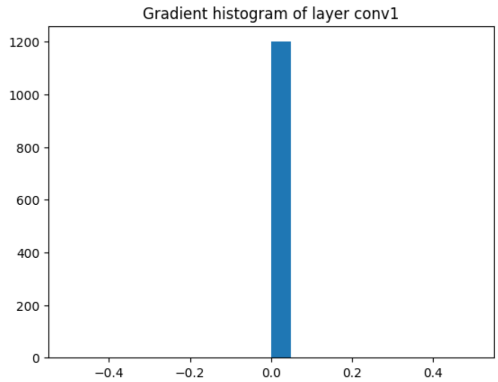


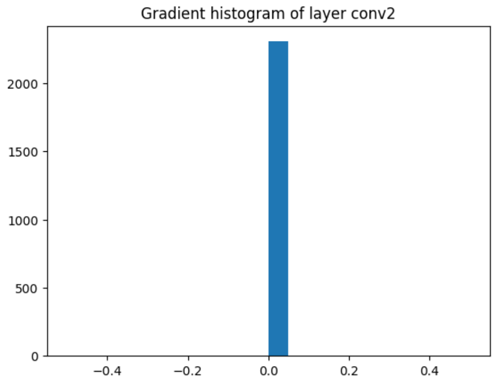


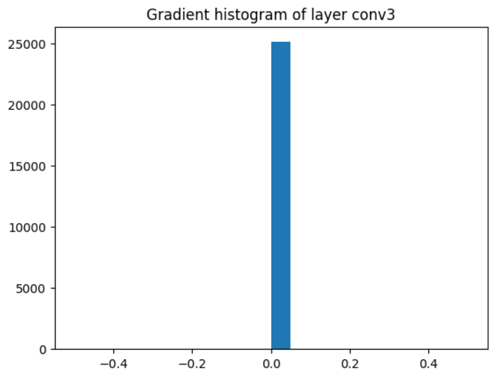


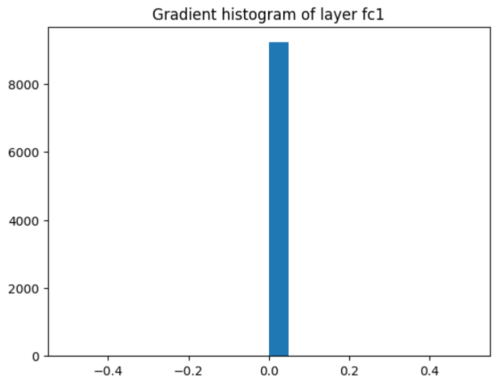


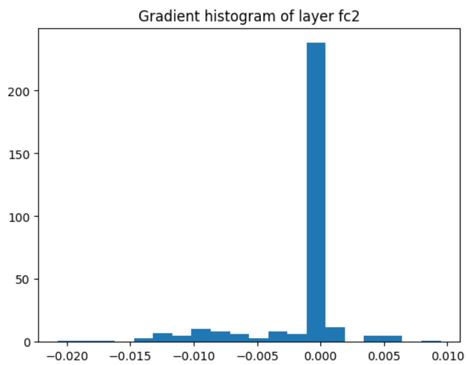
Zero Initialization:











In Convolutional Neural Networks (CNNs), the "dying ReLU" problem occurs when ReLU activation neurons become stuck at outputting 0 due to consistently negative weighted sums of their inputs. This issue can render a portion of the network 'dead,' leading to ineffective training as these neurons do not contribute to learning. It results in a loss of model capacity to capture intricate patterns in the data and can hinder the network's ability to adapt during training.