Name: **Zahra Soukhtedel** Assignment Number: **2**
Course: **Deep Learning** Date: March 30, 2023

Problem 1

1.1

Batch normalization helps to address the issue of exploding or vanishing gradients by normalizing the input to each layer. This means that the input values are scaled to have zero mean and unit variance, which helps to prevent extreme values that can cause the gradients to explode or vanish during backpropagation. By reducing the impact of extreme values, batch normalization allows for a larger learning rate to be used without causing instability in the training process. This is because a larger learning rate allows for faster convergence of the model, but can also cause the gradients to become unstable if not properly controlled. Batch normalization is a powerful technique that helps to improve the stability of deep neural networks, allowing for larger learning rates and faster convergence without sacrificing performance.
Batch normalization is all about keeping the activations of all layers normalized, preventing them from becoming too large or small. So this directly helps to prevent exploding/vanishing gradient. Due to this reason, batch normalization allows higher learning rates.

1.2

The effect of multiplying the pseudo weights in batch normalization by a fixed value α on the gradient with respect to the weights will result in a scaling of the gradients with respect to the weights by a factor of α. This means that the gradients with respect to the weights will be either amplified or reduced depending on the value of α. However, the gradient with respect to other network parameters such as the bias and scaling factors will not be affected by this multiplication since they are not directly influenced by the pseudo weights. Therefore, the effect on the gradient with respect these parameters will remain unchanged. Overall, the effect of multiplying the pseudo weights by a fixed value α on the gradient will depend on the specific architecture and parameters of the model, and should be carefully tuned and tested to ensure optimal performance.

1.3

Increasing the batch size in batch normalization will affect the regularization property because it changes the statistics used to normalize the input to each layer. With a larger batch size, the mean and variance used for normalization will be more representative of the entire dataset, rather than just a small subset of it. This means that the normalization will be more consistent across batches, which can reduce the amount of noise in the gradients and improve the stability of the training process. However, this also means that the regularization effect of batch normalization may be reduced, as the normalization is less likely to have a randomizing effect on the

input values. Therefore, increasing the batch size can lead to a trade-off between regularization and stability in the training process.

1.4

Batch normalization is conducted similarly in fully connected and convolutional networks, but there are variations in the calculation of statistics.
In fully connected networks, batch normalization is typically applied to the activations of each layer. The mean and variance of the activations are computed over the entire batch of training examples, and then utilized to normalize the activations before applying scale and shift parameters. This guarantees that the input to each layer is consistently centered and scaled, disregarding the input values' distribution.
In convolutional networks, batch normalization is typically applied to the outputs of each convolutional layer. The mean and variance are calculated separately for each channel of the output feature map and used to normalize the output before applying the scale and shift parameters. This ensures that the output of each channel is consistently centered and scaled, regardless of the features' spatial location.
Furthermore, batch normalization is frequently applied before the activation function in convolutional networks, while it is generally applied after the activation function in fully connected networks. This can affect the regularization effect of batch normalization since applying it before the activation function can prevent the activation from becoming too large or too small, reducing overfitting.
Despite some differences, batch normalization's fundamental idea is to normalize each layer's input or output, enhancing network stability and performance.

Problem 2

2.1

Advantages:
1. Increased receptive field: The dilated convolution filter can capture information from a larger area of the input without reducing the spatial resolution, which can be beneficial in tasks such as image segmentation or depth estimation.
2. Maintains spatial resolution: Unlike the extended convolution filter, the dilated convolution filter does not skip over any cells in the input, which means that it maintains the spatial resolution of the output.
Disadvantages:
1. Increased computational complexity: The dilated convolution filter requires more computations than a standard convolution filter, which can lead to slower training and inference times.
2. Limited dilation rates: The dilation rate of the filter is limited by the size of the input and the desired output size, which can limit its effectiveness in capturing information from very large areas of the input.
3. Limited applicability: The dilated convolution filter may not be suitable for all types of data or tasks, and its effectiveness may depend on the specific characteristics of the input and output data.

2.2

suppose that:
Dilated Convolution Layer 1: Dilation rate = d_1, Kernel size = $r_1 * c_1$
Dilated Convolution Layer 2: Dilation rate = d_2, Kernel size = $r_2 * c_2$
Dilated Convolution Layer 3: Dilation rate = d_3, Kernel size = $r_3 * c_3$
with no padding, the input range observed by element i,j in the output can be specified as follows:
For Dilated Convolution Layer 1:
The output element at position (i, j) in this layer observes a receptive field of size $(r_1 + (d_1 - 1) * (r_1 - 1)) * (c_1 + (d_1 - 1) * (c_1 - 1))$, centered at (i, j).
For Dilated Convolution Layer 2:
The output element at position (i, j) in this layer observes a receptive field of size $((r_1 + (d_1 - 1) * (r_1 - 1)) + (r_2 - 1) * (r_1 + (d_1 - 1) * (r_1 - 1) - 1)) * ((c_1 + (d_1 - 1) * (c_1 - 1)) + (c_2 - 1) * (c_1 + (d_1 - 1) * (c_1 - 1) - 1))$.
For Dilated Convolution Layer 3:
The output element at position (i,j) in this layer observes a receptive field of size $((r_1 + (d_1 - 1) * (r_1 - 1)) + (r_2 - 1) * (r_1 + (d_1 - 1) * (r_1 - 1) - 1) + (r_3 - 1) * ((r_1 + (d_1 - 1) * (r_1 - 1)) + (r_2 - 1) * (r_1 + (d_1 - 1) * (r_1 - 1) - 1) - 1)) * ((c_1 + (d_1 - 1) * (c_1 - 1)) + (c_2 - 1) * (c_1 + (d_1 - 1) * (c_1 - 1) - 1) + (c_3 - 1) * ((c_1 + (d_1 - 1) * (c_1 - 1)) + (c_2 - 1) * (c_1 + (d_1 - 1) * (c_1 - 1) - 1) - 1))$.
The receptive field size increases with each layer, enabling the network to capture larger spatial contexts in the input.

2.3

With parameters as follows
parameters: D, N, K, and I.
D represents the number of filters or feature maps in the convolutional layer.
N represents the dilation rate, which determines the spacing between the values in the kernel.
K represents the kernel size or filter size.
I represents the input size or dimensions of the input image.
we have:
The output of this network in terms of D, N, K, and I is

$$D * (I - (K - 1) * N) * (I - (K - 1) * N).$$

We first subtract (K-1)*N from the input size I, as this is the effective receptive field of the convolutional layer. We then multiply this value by D, as there are D filters in the layer.

Problem 3

3.1

based on parameters defined, we have:

$$z = \sum_{i=1}^{3} \sum_{j=1}^{3} A_{i,j} W_{3(i-1)+j}$$

And for each $A_{i,j}$ we have that is a maximum in a 2*2 block of P, as follows:

$$A_{i,j} = max\{P_{2i-1,2j-1}, P_{2i,2j-1}, P_{2i-1,2j}, P_{2i,2j}\}$$

So for $\frac{\partial loss}{\partial P_{i,j}}$ we have:

$$\frac{\partial loss}{\partial P_{i,j}} = \frac{\partial loss}{\partial z} * \frac{\partial z}{\partial P_{i,j}} = \frac{\partial loss}{\partial z} * \frac{\partial(\sum_{k=1}^{3}\sum_{l=1}^{3} A_{k,l} * W_{3(k-1)+l})}{\partial P_{i,j}}$$

$$= \frac{\partial loss}{\partial z} * \sum_{k=1}^{3}\sum_{l=1}^{3}(\frac{\partial(A_{k,l} * W_{3(k-1)+l})}{\partial P_{i,j}})$$

$$= \frac{\partial loss}{\partial z} * \sum_{k=1}^{3}\sum_{l=1}^{3} W_{(3(k-1)+l)}\frac{\partial A_{k,l}}{\partial P_{i,j}}$$

if $P_{i,j}$ be the maximum in its block, then $\frac{\partial A_{k,l}}{\partial P_{i,j}}$ for its corresponding $A_{k^*,l^*} = P_{i,j}$ would be one, and other parts would be zero, and if it is not maximum in its block, all parts in sum would be zero, so we have:

$$\frac{\partial loss}{\partial P_{i,j}} = \frac{\partial loss}{\partial z} * \sum_{k=1}^{3}\sum_{l=1}^{3} W_{(3(k-1)+l)}\frac{\partial A_{k,l}}{\partial P_{i,j}} = \begin{cases} \frac{\partial loss}{\partial z} * W_{(3(k^*-1)+l^*)} & \text{if its maximum} \\ 0 & \text{if its not maximum.} \end{cases}$$

3.2

First we know that for each i, j $P_{i,j}$ is calculated as follows:

$$P_{i,j} = \sum_{k=1}^{3}\sum_{l=1}^{3}\sum_{t=1}^{3} {W'}_{k,l,t} * x_{i+k-2,j+l-2,t}$$

Therefor we have:

$$\forall i,j = 1,...,6 : \frac{\partial P_{i,j}}{\partial {W'}_{s,u,v}} = x_{i+s-2,u+l-2,v}$$

And from the previous section we can compute value of $\frac{\partial loss}{\partial P_{i,j}}$, Therefor we can conclude that:

$$\forall i,j = 1,...,6 : \frac{\partial loss}{\partial {W'}_{s,u,v}} = \frac{\partial loss}{\partial P_{i,j}} * \frac{\partial P_{i,j}}{\partial {W'}_{s,u,v}} = \frac{\partial loss}{\partial P_{i,j}} * x_{i+s-2,u+l-2,v}$$

$$\frac{\partial loss}{\partial {W'}_{s,u,v}} = \begin{cases} \frac{\partial loss}{\partial z} * W_{(3(k^*-1)+l^*)} * x_{i+s-2,u+l-2,v} & \text{if its maximum} \\ 0 & \text{if its not maximum.} \end{cases}$$

Problem 4

4.1

No, the effect of SE block action in different depths is not the same. SE blocks have a greater effect on the network's performance when placed in earlier layers, Because earlier layers contain more general features that are important for the overall task, while later layers contain more specialized features that may not benefit as much from channel-wise attention. Also placing SE blocks in later layers may lead to overfitting due to the smaller number of channels.

4.2

In the formula r represents the reduction ratio, which determines the number of channels in the intermediate representation of the excitation unit. By decreasing the number of channels, the excitation unit can learn a more efficient representation of the input features and has the ability to enhance the network's performance through minimizing overfitting and enhancing generalization. if r is set too low, there is a risk of losing information and decreasing performance and if r is set too high, there is a risk of increased computational cost and overfitting. So the value of r should be carefully chosen based on the specific task and network architecture and we are using r in the dimensions of the weight matrix to control the complexity and computational cost of the excitation unit.

Problem 5

5.1

First network:
To compute the number of parameters in first network, we need to consider the number of parameters in each layer. Layer one parameters can be computed as follows:
Since this is a depth-wise convolutional layer, each channel in the input is convolved with its own filter, resulting in a total of 3 filters. The output size remains the same as the input size since we have used padding.
Number of parameters per filter = 3*3(filter size) + 1 (bias term) = 10.
Total number of parameters in layer one = (number of filters) * (number of parameters per filter) = 3 * 10 = 30.
The number of parameters in the second layer two can be computed as follows:
Total number of parameters in layer two = (number of filters) * (number of parameters per filter) = 128 * (3*1*1+ 1) = 512.
Therefore, **the total number of parameters in the MobielNet-V1 network is** 512 + 10 = **522**.
Second network:
To compute the number of parameters in first network we have:
Total number of parameters in the second network = (number of filters) * (number of parameters per filter) = 128 * (3*3*3+ 1) = **3584**.

5.2

To compute the number of parameters in this network, we need to consider the number of parameters in each layer.
Layer one contains 18 point-wise Conv filters with size 1*1*3, with input size 28*28*3 and output size of 28*28*18. Therefor we have:
Total number of parameters in layer one = (number of filters) * (number of parameters per filter) = 18 * (3*1*1+ 1) = 72.
Layer Two is a depth-wise Conv layer, each channel in the input is convolved with its own filter, resulting in a total of 18 filters. The output size remains the same as the input size since we have used padding.with input and output size 28*28*18. Therefor we have:
Number of parameters per filter = 3*3(filter size) + 1 (bias term) = 10.

Total number of parameters in layer two = (number of filters) * (number of parameters per filter) = 18 * 10 = 180.
Layer three contains 3 point-wise Conv filters with size 1*1*18, with input size 28*28*18 and output size of 28*28*3. Therefor we have:
Total number of parameters in layer three = (number of filters) * (number of parameters per filter) = 3 * (18*1*1+ 1) = 57.
So, **the total number of parameters in MobielNet-V2 network** is equal to 72 + 180 + 57 = **309**.
As we can see, the number of parameters has decreased from 522 to 309 in MobielNet-V2 compared to MobielNet-V1.

Problem 6

6.1

We have equations below:

$$a^l w^{l+1} + b^{l+1} = z^{l+1}$$
$$Act(z^{l+1}) = a^{l+1}$$
$$a^{l+1} w^{l+2} + b^{l+2} = z^{l+2}$$
$$z^{l+2} + a^l = z^{l+3}$$
$$Act(z^{l+3}) = a^{l+2}$$

To calculate $\frac{\partial a^{l+2}}{\partial a^l}$ based on chain rule we have:

$$\begin{aligned}
\frac{\partial a^{l+2}}{\partial a^l} &= \frac{\partial a^{l+2}}{\partial z^{l+3}} * \frac{\partial z^{l+3}}{\partial a^l} \\
&= Act'(z^{l+3}) \frac{\partial z^{l+3}}{\partial a^l} \\
&= Act'(z^{l+3})(\frac{\partial a^l}{\partial a^l} + \frac{\partial z^{l+2}}{\partial a^l}) \\
&= Act'(z^{l+3})(1 + w^{l+2} \frac{\partial a^{l+1}}{\partial a^l}) \\
&= Act'(z^{l+3})(1 + w^{l+2} \frac{\partial a^{l+1}}{\partial z^{l+1}} \frac{\partial z^{l+1}}{\partial a^l}) \\
&= Act'(z^{l+3})(1 + w^{l+2} Act'(z^{l+1}) \frac{\partial z^{l+1}}{\partial a^l}) \\
&= Act'(z^{l+3})(1 + w^{l+2} Act'(z^{l+1}) w^{l+1})
\end{aligned} \tag{1}$$

Therefore $\frac{\partial loss}{\partial a^l}$ with knowing $\frac{\partial loss}{\partial a^{l+2}}$ is as follows:

$$\begin{aligned}
\frac{\partial loss}{\partial a^l} &= \frac{\partial loss}{\partial a^{l+2}} * \frac{\partial a^{l+2}}{\partial a^l} \\
&= \frac{\partial loss}{\partial a^{l+2}} * Act'(z^{l+3})(1 + w^{l+2} Act'(z^{l+1}) w^{l+1}) \\
&= \frac{\partial loss}{\partial a^{l+2}} * Act'(z^{l+3}) + \frac{\partial loss}{\partial a^{l+2}} * Act'(z^{l+3}) w^{l+2} Act'(z^{l+1}) w^{l+1}
\end{aligned} \tag{2}$$

6.2

This is because the bypass edge allows the gradient to flow directly from the input to the output without passing through multiple layers. By adding a bypass edge, the network can more easily learn to identify and preserve important features in the input data, even as it passes through multiple layers of the MLP. This can help to prevent the gradient from becoming too small and ensure that the network is able to learn and adapt to new data.
In two layer MLP, without residual bypass we have:

$$\begin{aligned}\frac{\partial loss}{\partial a^l} &= \frac{\partial loss}{\partial a^{l+2}} * \frac{\partial a^{l+2}}{\partial a^l} \\ &= \frac{\partial loss}{\partial a^{l+2}} * Act'(z^{l+3})(w^{l+2}Act'(z^{l+1})w^{l+1}) \\ &= \frac{\partial loss}{\partial a^{l+2}} * Act'(z^{l+3})w^{l+2}Act'(z^{l+1})w^{l+1}\end{aligned} \tag{3}$$

The extra term $\frac{\partial loss}{\partial a^{l+2}} * Act'(z^{l+3})$ in resnet architecture can be used to prevent the gradient from shrinking through more layers of the network. The additional $\frac{\partial loss}{\partial a^{l+2}} * Act'(z^{l+3})$ term in Resnet architecture compared to MLP prevents the gradient from shrinking through more layers of the network.