Solution for Assignment 2 - Deep Learning Course

Submitted by: Yarden Cohen

Question 1

The equation

$$z_i = w_1 x_{i-1} + w_2 x_i + w_3 x_{i+1}$$

defines a 1D convolution with a kernel size of three, stride of one, and dilation zero. Write out the equation for a 1D convolution with a kernel size of seven, a dilation rate of two, and a stride of three.

Solution:

Definitions:

- **Kernel size**: The number of elements in the kernel. For a kernel size of 7, the kernel has 7 elements: $w_1, w_2, \ldots, w_7$.
- **Dilation rate**: Determines the spacing between elements in the kernel. A dilation rate of 2 means that there is a gap of 1 element between the elements in the input that the kernel is applied to.
- **Stride**: The step size with which we move the kernel across the input. A stride of 3 means that the kernel moves 3 positions at a time.

The general form of the equation, using 1-based indexing, is:

$$z_k = \sum_{i=1}^{7} w_i \cdot x_{s+(i-1)(d+1)}$$

Where:

- k is the output index (starting at 1)
- s is the starting input index for each output
- d is the dilation rate (2 in this case)

With a stride of 3 and 1-based indexing, the starting index s for each output k is:

$$s = 1 + 3(k-1)$$

Expanding the sum and substituting $s = 1 + 3(k-1)$ and $d = 2$, we get:

$$\begin{aligned} z_k = w_1 \cdot x_{1+3(k-1)} +& \\ w_2 \cdot x_{1+3(k-1)+3} +& \\ w_3 \cdot x_{1+3(k-1)+6} +& \\ w_4 \cdot x_{1+3(k-1)+9} +& \\ w_5 \cdot x_{1+3(k-1)+12} +& \\ w_6 \cdot x_{1+3(k-1)+15} +& \\ w_7 \cdot x_{1+3(k-1)+18}& \end{aligned}$$

Final equation:

$$z_k = w_1 x_{1+3(k-1)} + w_2 x_{4+3(k-1)} + w_3 x_{7+3(k-1)} + w_4 x_{10+3(k-1)} + w_5 x_{13+3(k-1)} + w_6 x_{16+3(k-1)} + w_7 x_{19+3(k-1)}$$

Question 2

A network consists of three 1D convolutional layers. At each layer, a zero-padded convolution with kernel size seven, stride one, and dilation zero is applied. What size is the receptive field of hidden units in the third hidden layer?

Solution:

To calculate the receptive field size of hidden units in the third hidden layer of a network with three 1D convolutional layers, each using a zero-padded convolution with kernel size seven, stride one, and dilation zero, we can follow the standard formula for computing the receptive field of deep convolutional layers.

Formula for the Receptive Field

The receptive field size RF of a layer in a convolutional network can be recursively calculated as:

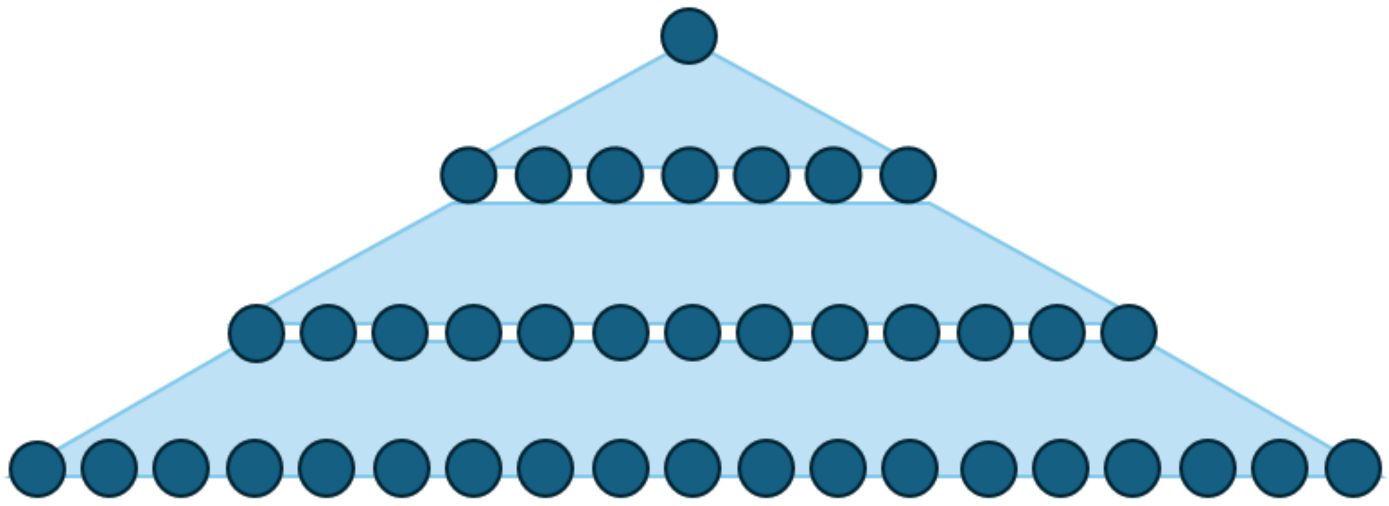
$$RF_n = RF_{n-1} + (K-1) \times \prod_{i=1}^{n-1} S_i$$

Where:

- RF_n is the receptive field of the n-th layer.
- RF_{n-1} is the receptive field of the $(n-1)$-th layer.
- K is the kernel size of the n-th layer.
- S_i is the stride of the i-th layer up to $n-1$.

Calculation Steps

1. **First Layer**
   - Kernel Size, $K = 7$
   - Stride, $S = 1$
   - Receptive Field, $RF_1 = 7$ (because the receptive field is just the kernel size for the first layer when the stride is one).
2. **Second Layer**
   - Continuing from the first layer's receptive field:
   - $RF_2 = RF_1 + (K-1) \times 1 = 7 + (7-1) \times 1 = 7 + 6 = 13$
3. **Third Layer**
   - $RF_3 = RF_2 + (K-1) \times 1 = 13 + (7-1) \times 1 = 13 + 6 = 19$



Conclusion

The receptive field of hidden units in the third hidden layer is 19 units. This means each unit in the third hidden layer is affected by 19 input units from the original input data.

Question 3

Consider a 1D convolutional network where the input has three channels. The first hidden layer is computed using a kernel size of three and has four channels. The second hidden layer is computed using a kernel size of five and has ten channels. How many biases and how many weights are needed for each of these two convolutional layers?

Solution:

Layer 1:

- **Kernel Size**: 3
- **Input Channels**: 3
- **Output Channels**: 4
- **Weights Needed**: 36 (4 output channels * 3 input channels * 3 kernel size)
- **Biases Needed**: 4 (one per output channel)

Layer 2:

- **Kernel Size**: 5
- **Input Channels**: 4
- **Output Channels**: 10
- **Weights Needed**: 200 (10 output channels * 4 input channels * 5 kernel size)
- **Biases Needed**: 10 (one per output channel) on:

Total number of parameters: $36 + 4 + 200 + 10 = 250$

Question 4

Consider a convolutional residual block that contains a batch normalization operation, followed by a ReLU activation function, and then a 3×3 convolutional layer. If the input and output both have 512 channels, how many parameters are needed to define this block?

Solution

Consider the parameters in each component of the block:

Components of the Residual Block:

1. **Batch Normalization**:
    - For each channel, batch normalization has two parameters: a scale parameter γ and a shift parameter β.
    - Since there are 512 channels, the total number of parameters for batch normalization is:
    $$\text{Parameters for Batch Normalization} = 2 \times 512 = 1024$$
2. **ReLU Activation**:
    - The ReLU activation function is a non-parametric operation, meaning it does not add any additional parameters.
3. 3×3 **Convolutional Layer**:
    - The convolutional layer has a kernel size of 3×3 and 512 input channels, with 512 output channels.
    - **Input Channels**: These correspond to the depth of the input feature map. In this case, there are 512 input channels, which means the input feature map has a depth of 512.
    - **Output Channels**: These correspond to the number of kernels (or filters) applied to the input. Each kernel generates one output channel. So, with 512 output channels, there are 512 different kernels.
    - The number of parameters for the convolutional layer is calculated as:
    $$\text{Parameters for Convolutional Layer} = \\ (\text{kernel height} \times \text{kernel width} \times \text{input channels} \times \text{output channels}) + \text{biases}$$
    - Since we assume a bias term for each output channel, the total number of parameters for the convolutional layer is:
    $$(3 \times 3 \times 512 \times 512) + 512 = 2359296 + 512 = 2359808$$

Total Parameters in the Block:

Adding up the parameters for batch normalization and the convolutional layer:

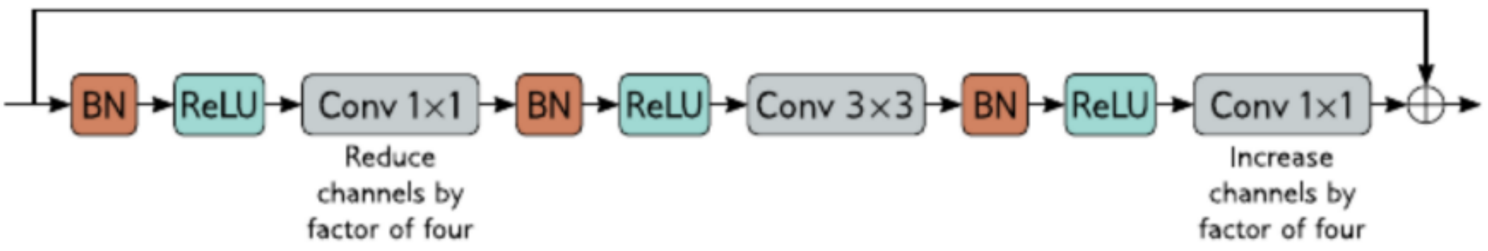
$$\text{Total Parameters} = 1024 + 2359808 = 2360832$$

Final Answer

The total number of parameters needed to define this convolutional residual block is **2,360,832**.

Question 5

Now consider a bottleneck residual block that contains three batch normalization/ReLU/convolution sequences. The first uses a 1×1 convolution to reduce the number of channels from 512 to 128. The second uses a 3×3 convolution with the same number of input and output channels. The third uses a 1×1 convolution to increase the number of channels from 128 to 512. How many parameters are needed to define this block?


BN
ReLU
Conv 1×1
Reduce channels by factor of four
BN
ReLU
Conv 3×3
BN
ReLU
Conv 1×1
Increase channels by factor of four

Solution:

1. **First Sequence (1×1 Convolution from 512 to 128 channels)**:
   - **Batch Normalization**:
     - Parameters: γ and β for each of the 512 input channels.
     - Total: $2 \times 512 = 1024$

- **1×1 Convolution**:
   - Each 1×1 convolution filter processes all 512 input channels at once, resulting in 1×1×512 weights per filter.
   - To produce 128 output channels, we need 128 such filters.
   - Weights: $1 \times 1 \times 512 \times 128 = 65,536$
   - Biases: 128 (one bias per output channel)
   - Total parameters: $65,536 + 128 = 65,664$

2. **Second Sequence (3×3 Convolution with 128 input and output channels)**:
   - **Batch Normalization**:
     - Parameters: γ and β for each of the 128 channels.
     - Total: $2 \times 128 = 256$
   - **3×3 Convolution**:
     - Each 3×3 convolution filter processes all 128 input channels.
     - Weights: $3 \times 3 \times 128 \times 128 = 147,456$
     - Biases: 128
     - Total: $147,456 + 128 = 147,584$

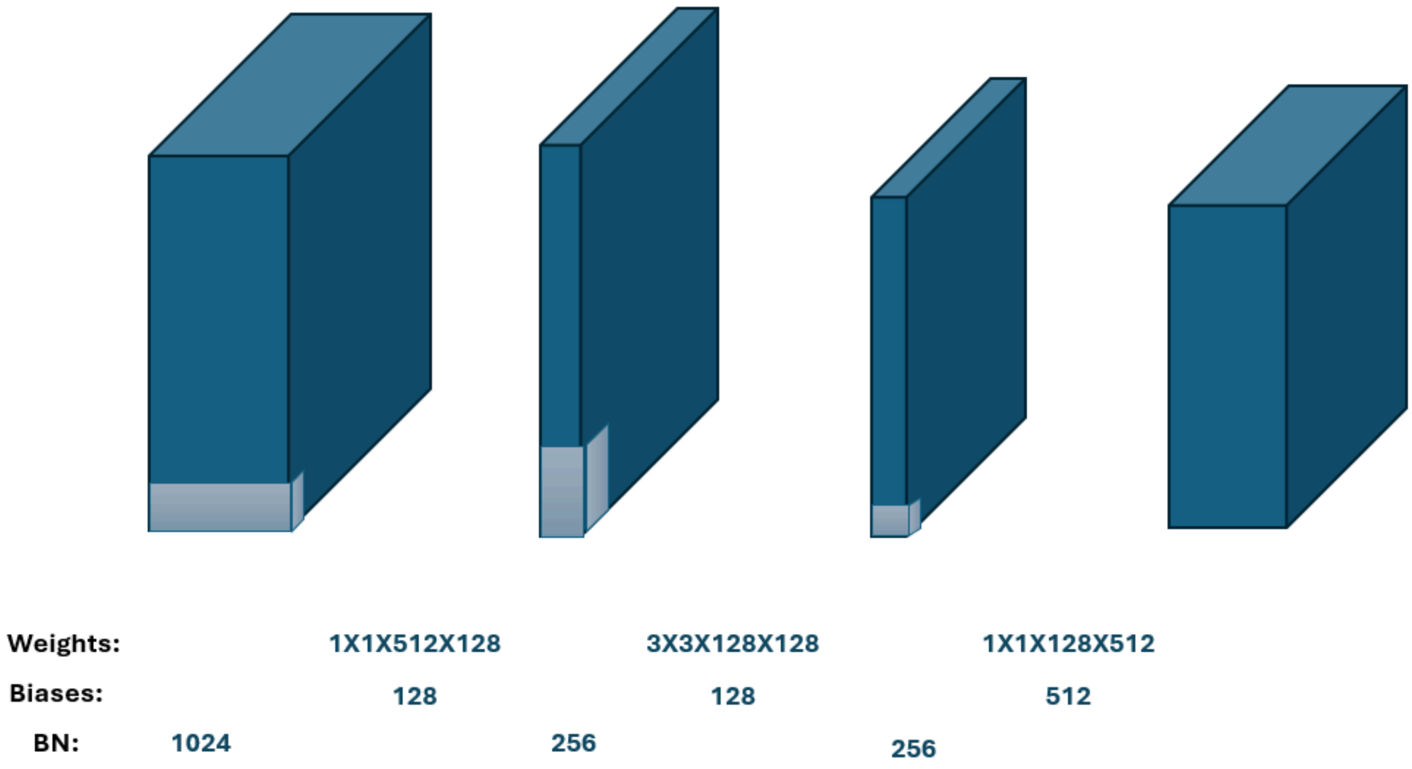
3. **Third Sequence (1×1 Convolution to increase channels from 128 to 512)**:

- **Batch Normalization**:
    - Parameters: γ and β for each of the 128 channels.
    - Total: $2 \times 128 = 256$
- **1×1 Convolution**:
    - Weights: $1 \times 1 \times 128 \times 512 = 65,536$
    - Biases: 512
    - Total: $65,536 + 512 = 66,048$

Summing All Parameters:

$$\text{Total parameters} = \text{First sequence} + \text{Second sequence} + \text{Third sequence}$$
$$= (1024 + 65,664) + (256 + 147,584) + (256 + 66,048)$$
$$= 280,832$$

The total number of parameters needed to define this bottleneck residual block is **280,832**.



Question 6

Consider a self-attention computation

$$Sa[X] = V \cdot \text{Softmax}(K^T \cdot Q)$$

where

$$V = b_v 1^T + W_v X$$
$$K = b_k 1^T + W_k X$$
$$Q = b_q 1^T + W_q X$$

that processes N inputs of length D ($X \in \mathbb{R}^{D \times N}$) to produce N outputs of the same size. How many weights and biases are used to compute the queries, keys, and values? How many weights and biases would there be in a fully connected network relating all DN inputs to all DN outputs?

Solution:

Self-Attention Weights and Biases

Definitions:

Given:

$$V = b_v \mathbf{1}^T + W_v X$$
$$K = b_k \mathbf{1}^T + W_k X$$
$$Q = b_q \mathbf{1}^T + W_q X$$

where:

- $X \in \mathbb{R}^{D \times N}$: Input matrix with N inputs, each of length D.
- $W_v, W_k, W_q \in \mathbb{R}^{D \times D}$: Weight matrices for values, keys, and queries.
- $b_v, b_k, b_q \in \mathbb{R}^D$: Bias vectors for values, keys, and queries.
- $\mathbf{1}^T \in \mathbb{R}^{1 \times N}$: Row vector of ones to broadcast the bias vectors across all N inputs.

Weights and Biases to compute the queries, keys, and values:

- **Weights** W_v, W_k, W_q: Each is a $D \times D$ matrix.
  - Number of weights per matrix: $D \times D$
  - Total weights for all three matrices: $3 \times D \times D$
- **Biases** b_v, b_k, b_q: Each is a D-dimensional vector.
  - Number of biases per vector: D
  - Total biases for all three vectors: $3 \times D$

Therefore, the total number of weights and biases for the self-attention mechanism are:

$$\text{Total weights} = 3 \times D \times D$$
$$\text{Total biases} = 3 \times D$$
$$\text{Total number of parameters} = 3 \times D \times (D + 1)$$

Fully Connected Network Weights and Biases

For a fully connected network that relates all DN inputs to all DN outputs:

- **Inputs**: DN
- **Outputs**: DN
- **Weights**: Each input is connected to each output.
    - Number of weights: $(DN) \times (DN) = D^2N^2$
- **Biases**: One bias per output.
    - Number of biases: DN

Comparison

1. **Self-Attention Weights and Biases**:
    - Total weights: $3D^2$
    - Total biases: $3D$
2. **Fully Connected Network Weights and Biases**:
    - Total weights: D^2N^2
    - Total biases: DN

Question 7

Show that the self-attention operation

$$Sa[X] = V \cdot \mathrm{Softmax}(K^T \cdot Q)$$

is equivariant to a permutation XP of the data X, where P is a permutation matrix. In other words, show that $Sa[XP] = Sa[X]P$.

Solution

Permutation Matrix

A **permutation matrix** P is a square binary matrix with exactly one entry of 1 in each row and each column, with all other entries being 0. It rearranges the rows or columns of another matrix when multiplied:

- Multiplying from the left permutes the rows.
- Multiplying from the right permutes the columns.

Properties of a Permutation Matrix:

1. **Orthogonality**: $P^T = P^{-1}$. Multiplying by P and then P^T (or vice versa) yields the identity matrix.
2. **Row and Column Operations**: Multiplying a matrix by P from the right permutes the columns.

Definitions

- $X \in \mathbb{R}^{D\times N}$: Input matrix with D features and N sequence items.
- $P \in \mathbb{R}^{N\times N}$: Permutation matrix.
- $V = b_v 1^T + W_v X$
- $K = b_k 1^T + W_k X$
- $Q = b_q 1^T + W_q X$
- $W_v, W_k, W_q \in \mathbb{R}^{D\times D}$: Weight matrices.
- $b_v, b_k, b_q \in \mathbb{R}^D$: Bias vectors.
- $1^T \in \mathbb{R}^N$: Row vector of all ones.

Proof

Given XP as input, the transformed outputs Q', K', and V' are derived as follows:

For Q' (similar for K' and V'):

$$\begin{aligned} Q' &= b_q 1^T + W_q(XP) \\ &= b_q 1^T + (W_q X)P \\ &= (b_q 1^T)P + (W_q X)P \\ &= (b_q 1^T + W_q X)P \\ &= QP \end{aligned}$$

- $b_q 1^T = (b_q 1^T)P$ holds because permuting identical columns (all columns of $b_q 1^T$ are the vector b_q) does not change their arrangement.

Thus, $K' = KP$ and $V' = VP$ can be similarly shown.

Substituting Into the Self-Attention Formula:

$$\begin{aligned} Sa[XP] &= V' \cdot \mathrm{Softmax}[(K')^T \cdot Q'] \\ &= (VP) \cdot \mathrm{Softmax}[(KP)^T \cdot (QP)] \\ &= (VP) \cdot \mathrm{Softmax}[P^T \cdot K^T \cdot Q \cdot P] \end{aligned}$$

Using the Properties: 1. $P^T = P^{-1}$ for permutation matrices. **2.** Softmax is invariant to permutation matrices.

$$\begin{aligned} Sa[XP] &= (VP) \cdot \mathrm{Softmax}[P^T \cdot K^T \cdot Q \cdot P] \\ &= (VP) \cdot \mathrm{Softmax}[P^{-1} \cdot K^T \cdot Q \cdot P] && (1) \\ &= (VP) \cdot P^{-1} \cdot \mathrm{Softmax}[K^T \cdot Q] \cdot P && (2) \\ &= V(P \cdot P^{-1}) \cdot \mathrm{Softmax}[K^T \cdot Q] \cdot P \\ &= V \cdot \mathrm{Softmax}[K^T \cdot Q] \cdot P && (\ P \cdot P^{-1} = I) \\ &= Sa[X]P && (Sa[X] = V \cdot \mathrm{Softmax}[K^T \cdot Q]) \end{aligned}$$

This proof confirms that $Sa[XP] = Sa[X]P$, demonstrating the permutation equivariance of the self-attention mechanism.

Question 8

Consider transforming a standard normal distribution

$$p(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$$

with the function

$$x = f[z] = \frac{1}{1+e^{-z}}$$

Find an expression for the transformed distribution $p(x)$.

Solution

Standard Normal Distribution

The probability density function (PDF) of a standard normal distribution is defined as:

$$p_Z(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}, \quad -\infty < z < \infty$$

- This expression provides the likelihood of z occurring within the normal distribution.

Change of Variables Theorem

For a continuous random variable Z with PDF $p_Z(z)$ and a transformation defined by a monotonic function $X = f(Z)$:

$$p_X(x) = p_Z(f^{-1}(x)) \left| \frac{d}{dx} f^{-1}(x) \right|$$

- This theorem allows us to determine the new PDF $p_X(x)$ after Z has been transformed by f, incorporating the inverse function $f^{-1}(x)$ and its derivative.

Inverse Function

The aim is to find the inverse of the logistic transformation:

$$x = f(z) = \frac{1}{1+e^{-z}}$$

Derivation of the inverse function:

$$\begin{aligned} x(1+e^{-z}) &= 1 \\ e^{-z} &= \frac{1-x}{x} \\ -z &= \ln\left(\frac{1-x}{x}\right) \\ z &= \ln\left(\frac{x}{1-x}\right) = \ln(x) - \ln(1-x) \end{aligned}$$

- This inverse function $f^{-1}(x)$ is derived to express z in terms of x.

Derivative of the Inverse Function

$$\begin{aligned}\frac{d}{dx}f^{-1}(x) &= \frac{d}{dx}[\ln(x) - \ln(1-x)] \\ &= \frac{1}{x} + \frac{1}{1-x} \\ &= \frac{1-x+x}{x(1-x)} \\ &= \frac{1}{x(1-x)}\end{aligned}$$

- This derivative adjusts the density $p_Z(z)$ to account for changes in variable space caused by the transformation, ensuring that $p_X(x)$ remains a valid probability density function (PDF).

Applying the Change of Variables Theorem

Combining the derived expressions to apply the theorem:

$$\begin{aligned}p_X(x) = p_Z(f^{-1}(x)) \left|\frac{d}{dx}f^{-1}(x)\right| \\ &= \frac{1}{\sqrt{2\pi}}e^{-\frac{(\ln(x)-\ln(1-x))^2}{2}} \cdot \left|\frac{1}{x(1-x)}\right| \\ &= \frac{1}{\sqrt{2\pi}}e^{-\frac{(\ln(x)-\ln(1-x))^2}{2}} \cdot \frac{1}{x(1-x)} \\ &= \frac{1}{\sqrt{2\pi}x(1-x)}e^{-\frac{(\ln(x)-\ln(1-x))^2}{2}}\end{aligned}$$

- Since the derivative $\frac{1}{x(1-x)}$ remains positive throughout the interval $0 < x < 1$, the absolute value does not alter the expression and therefore can be removed.

Final Result

The PDF of the transformed random variable X is:

$$p_X(x) = \frac{1}{\sqrt{2\pi}x(1-x)}e^{-\frac{(\ln(x)-\ln(1-x))^2}{2}}, \quad 0 < x < 1$$

This result shows the probability density of x after applying a logistic transformation to a standard normal variable z, fully accounting for the nonlinear transformation effects.

Question 9

We run the stochastic gradient descent algorithm for 1,000 iterations on a dataset of size 100 with a batch size of 20. For how many epochs did we train the model?

Solution

1. **Batches per Epoch**:
$$\text{Batches per Epoch} = \frac{\text{Dataset Size}}{\text{Batch Size}}$$
$$= \frac{100}{20} = 5$$
This indicates that each epoch consists of 5 batches.

2. **Total Epochs**:
$$\text{Total Epochs} = \frac{\text{Total Iterations}}{\text{Batches per Epoch}}$$
$$= \frac{1000}{5} = 200$$

Therefore, the model was trained for **200 epochs** over the course of 1,000 iterations.

Question 10

Stochastic gradient descent with momentum updates the model parameters ϕ as:

$$m_{t+1} = \beta \cdot m_t + (1-\beta)\sum_{i\in B_t}\frac{\partial l_i[\phi_t]}{\partial \phi}$$
$$\phi_{t+1} = \phi_t - \alpha \cdot m_{t+1}$$

Show that the momentum term m_t is a weighted sum of the gradients at the previous iterations and derive an expression for the coefficients (weights) of that sum.

Solution

Let $g_t = \sum_{i\in B_t}\frac{\partial l_i[\phi_t]}{\partial \phi}$ represent the gradient at time t.

The momentum update can be rewritten as:

$$m_{t+1} = \beta \cdot m_t + (1-\beta)\cdot g_t$$

Expanding recursively:

$$\begin{gathered} m_{t+1} = \beta \cdot m_t + (1-\beta)\cdot g_t \\ = \beta\cdot(\beta\cdot m_{t-1} + (1-\beta)\cdot g_{t-1}) + (1-\beta)\cdot g_t \\ = \beta^2\cdot m_{t-1} + \beta(1-\beta)\cdot g_{t-1} + (1-\beta)\cdot g_t \\ = \beta^3\cdot m_{t-2} + \beta^2(1-\beta)\cdot g_{t-2} + \beta(1-\beta)\cdot g_{t-1} + (1-\beta)\cdot g_t \end{gathered}$$

Generalizing for $t+1$ expansions:

$$m_{t+1} = \beta^{t+1}\cdot m_0 + (1-\beta)\sum_{k=0}^{t}\beta^k\cdot g_{t-k}$$

Assuming $m_0 = 0$:

$$m_{t+1} = (1-\beta)\sum_{k=0}^{t}\beta^k\cdot g_{t-k}$$

From this expression, we can see that m_{t+1} is indeed a weighted sum of the gradients at previous iterations.

The weight for the gradient at iteration $t-k$ is:

$$w_k = (1-\beta)\beta^k$$

where $k \in [0, t]$.