Machine Learning Group
Faculty IV – Electrical Engineering and Computer Science
Technische Universität Berlin

Exercise Sheet 6

Exercise 1: Fusing convolution and BatchNorm (10+15 P)

One of the tricks that helps improve the efficiency of training neural networks is batch normalization (BatchNorm; Ioffe and Szegedy, 2015). In the following, we focus on the case that we only have one spatial dimension (e.g. time steps). Denote[1] $\boldsymbol{z} = (z_j)_{j=1}^{d} \in \mathbb{R}^d$ to be an activation vector at a particular location (of total T spatial locations) in the feature map $Z \in \mathbb{R}^{d \times T}$. BatchNorm performs channel-wise

$$\hat{z}_j = \gamma_j \frac{(z_j - \mu_j)}{\sigma_j} + \beta_j,$$

where $\mu_j \in \mathbb{R}$ and $\sigma_j \in \mathbb{R}_+$ are mean and standard deviation—often estimated from samples in the mini-batch and running-average—and $\gamma_j \in \mathbb{R}$ and $\beta_j \in \mathbb{R}$ are trainable parameters. Because of the channel-wise computation, we can express BatchNorm in a matrix form

$$\hat{\boldsymbol{z}} \leftarrow \Gamma \boldsymbol{z} + \boldsymbol{\alpha},$$

where $\Gamma \in \mathbb{R}^{d \times d}$ and $\boldsymbol{\alpha} \in \mathbb{R}^d$.

(a) *Derive* the following entries of Γ and $\boldsymbol{\alpha}$:

$$\begin{aligned} \forall j \in \{1, \dots, d\} : \Gamma_{jj} &= \_\_ \in \mathbb{R} \\ \alpha_j &= \_\_ \in \mathbb{R} \\ \forall j \neq j' : \Gamma_{jj'} &= \_\_ \in \mathbb{R} \end{aligned}$$

Solution:
We first write

$$\begin{aligned} \hat{z}_j &= \frac{\gamma_j}{\sigma_j}(z_j - \mu_j) + \beta_j \\ &= \frac{\gamma_j}{\sigma_j} z_j - \frac{\gamma_j}{\sigma_j}\mu_j + \beta_j \\ &= \frac{\gamma_j}{\sigma_j} z_j + \left(\beta_j - \frac{\gamma_j}{\sigma_j}\mu_j\right). \end{aligned}$$

The derivation above implies that

$$\Gamma_{jj} = \gamma_j/\sigma_j, \quad \Gamma_{jj'} = 0, \qquad \alpha_j = \beta_j - \frac{\gamma_j}{\sigma_j}\mu_j.$$

Or, equivalently

$$\Gamma = \begin{bmatrix} \gamma_1/\sigma_1 & & \\ & \ddots & \\ & & \gamma_d/\sigma_d \end{bmatrix}, \quad \boldsymbol{\alpha} = \begin{bmatrix} \beta_1 - \frac{\gamma_1\mu_1}{\sigma_1} \\ \vdots \\ \beta_d - \frac{\gamma_d\mu_d}{\sigma_d} \end{bmatrix}.$$

Based on the matrix formulation, one can view the computation of BatchNorm as a convolution Φ_{BN} (i.e. $k_{\text{BN}} = 1$) layer on the feature map A with weights Γ and bias $\boldsymbol{\alpha}$. Commonly, the layer proceeds a convolution layer. Let Φ be such a convolution layer with kernel size k and $W \in \mathbb{R}^{d \times d_0 \times k}$ and $\boldsymbol{b} \in \mathbb{R}^d$ to be its weights and bias. Denote $A \in \mathbb{R}^{d_0 \times T}$ to be the input feature map to Φ and assume that the convolution layer Φ maintains the spatial size. We therefore have

$$Z = \Phi(A) \in \mathbb{R}^{d \times T}$$

[1] The notation $(z_j)_{j=1}^{d}$ represents a d-dimensional vector $(z_1, z_2, \dots, z_d)$.

where $\forall m \in \{1, \ldots, T\} : Z_{j,m} = (A \star W_{j,:,:})_m + b_j$ and the convolution operator $\star$ (as defined in Lecture Slide 15)

$$(A \star W_{j,:,:})_m = \sum_{\tau=1}^{k} \sum_{i=1}^{d_0} A_{i,m+(\tau-1)} W_{j,i,\tau}$$

Under this structure, one can fuse the convolution and BatchNorm layers together

$$\hat{Z} = \Phi_{\text{BN}}(\Phi(A)) = \Phi_F(A),$$

where Φ_F is a convolution layer with weights $\Lambda \in \mathbb{R}^{d \times d_0 \times k}$ and biases[2] $\boldsymbol{\beta} \in \mathbb{R}^d$.

(b) *Derive* the following quantities $\forall j \in \{1, \ldots, d\}$

$$\Lambda_{j,:,:} = \_\_\_ \in \mathbb{R}^{d_0 \times k}$$
$$\beta_j = \_\_\_ \in \mathbb{R}$$

Solution
Using (a), we have

$$\hat{Z}_{j,m} = \Gamma_{jj} Z_{j,m} + \alpha_j.$$

Replacing $Z_{j,m}$ with its convolution definition yields

$$\begin{aligned}
\hat{Z}_{j,m} &= \Gamma_{jj} \left[\left(\sum_{\tau=1}^{k} \sum_{i=1}^{d_0} A_{i,m+(\tau-1)} W_{j,i,\tau} \right) + b_j \right] + \alpha_j \\
&= \left(\sum_{\tau=1}^{k} \sum_{i=1}^{d_0} A_{i,m+(\tau-1)} (\Gamma_{jj} W_{j,i,\tau}) \right) + (\Gamma_{jj} b_j + \alpha_j) \\
\implies \Lambda_{j,:,:} &= \Gamma_{jj} W_{j,:,:} \\
\beta_j &= \Gamma_{jj} b_j + \alpha_j.
\end{aligned}$$

Exercise 2: Calculating the Size of Receptive Fields (10+10+10+45 P)

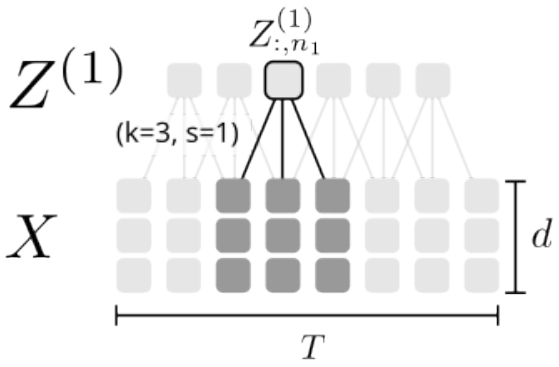

Figure 1: One-Dimensional convolution

Let $W^{(1)} \in \mathbb{R}^{d_1 \times d \times k}$ be a convolution weights with kernel size k and output dimension d_1. Let $X \in \mathbb{R}^{d \times T}$ be a feature map representing a sequence of T d-dimensional vectors. Recall the definition of convolution of W on the feature map X

$$\begin{aligned}
Z^{(1)}_{j,n_1} &= (X \star W^{(1)}_{j,:,:})_{n_1} \\
&= \sum_{\tau=1}^{k} \sum_{i=1}^{d} X_{i,n_1+\tau-1} W_{j,i,\tau},
\end{aligned}$$

[2] This $\boldsymbol{\beta}$ here is a different variable from what defined in the definition of BatchNorm in (a).

The receptive field of $Z^{(1)}_{:,n_1}$ is part of the input X that $Z^{(1)}_{:,n_1}$ depends on (See Figure 1). Trivially, in the case of one layer, the receptive field size of each $Z^{(1)}_{:,n_1}$ is k, and, it does not depend on the value of stride used s.

(a) Suppose we have a sequence of two convolutions with weights $W^{(1)}$ and $W^{(2)} \in \mathbb{R}^{d_2 \times d_1 \times k_2}$ with the values of stride s_1 and s_2 respectively. Denote Φ_1 and Φ_2 to be these two convolution operators respectively. We write

$$Z^{(1)} = X \star \Phi_1 \in \mathbb{R}^{d_1 \times T_1} \tag{1}$$

$$Z^{(2)} = Z^{(1)} \star \Phi_2 \in \mathbb{R}^{d_2 \times T_2} \tag{2}$$

Complete the connection diagrams in Figure 2 and fill in the corresponding receptive field sizes of $Z^{(2)}_{:,n_2}$ in Table 1.

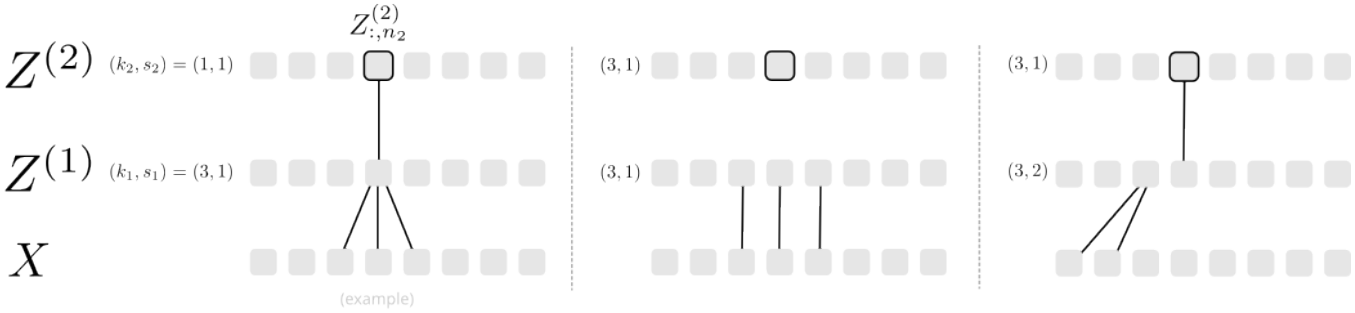

Figure 2: Draw connections between neurons in different layer according to different values of k_1, s_1, k_2, and s_2. It is sufficient to draw only connections that is relevant to $Z^{(2)}_{:,n_2}$.

(k_1, s_1)	(k_2, s_2)	Receptive Field Size of $Z^{(2)}_{:,n_2}$
(3, 1)	(1, 1)	3
(3, 1)	(3, 1)	___
(3, 2)	(3, 1)	___

Table 1: Based on the connections drawn in Figure 2, calculate the size of the receptive field in each setting.

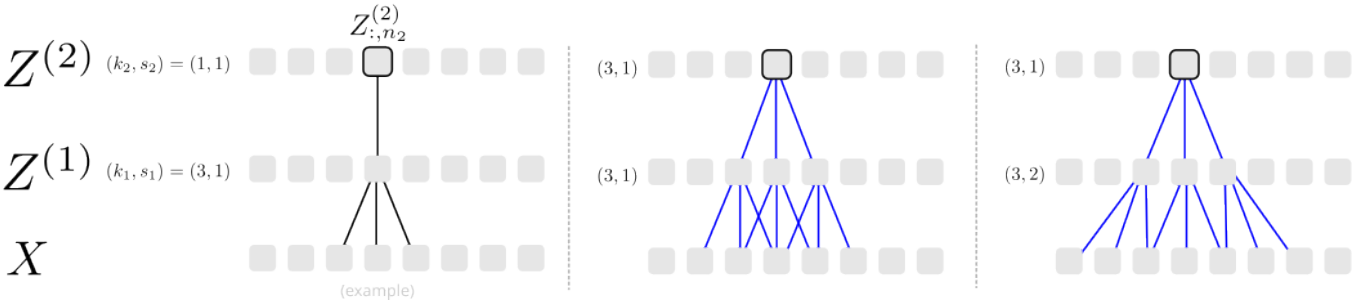

Figure 3: Solution of Figure 2. The size of each case's receptive field is $3, 5$, and 7 respectively.

(b) Let L is the number of convolution and pooling layers. Suppose these layers are stacked sequentially and denote $\{(k_l, s_l\}_{l=1}^{L}$ be the set of each layer's kernel size and stride. Araujo et al. (2019) show that the size of the top layer neuron $Z^{(L)}_{:,n_L}$'s receptive field can be computed from the kernel sizes and stride values via

$$\texttt{RF}(Z^{(L)}_{:,n_L}) = \sum_{l=1}^{L} \left((k_l - 1) \prod_{i=1}^{l-1} s_i \right) + 1.$$

Consider the architecture of VGG16 in Figure 4. *Compute* the receptive field sizes of neurons in these convolution layers: Layer 2, 4, 14, and 21. (Hint: It is sufficient to provide a few steps of the calculations and the numbers.)

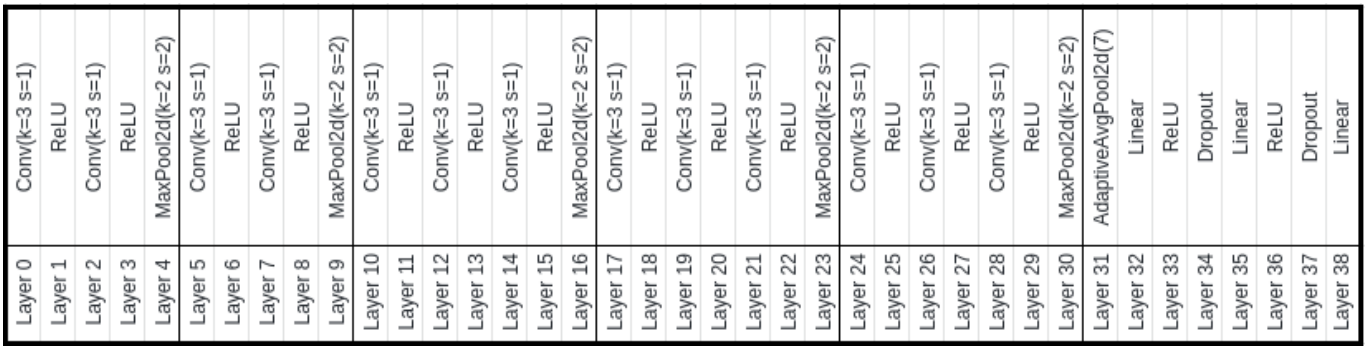

Figure 4: VGG16 Architecture

Solution:
The answers are 5, 6, 40, and 92 respectively.

(c) In many practical settings, deep learning models are complex and contain other types of layers that could influence the size of the receptive field; therefore, the formula above might not be suitable. Let $n_L \in T_L$ be the spatial location n_L after the L-th convolution. Alternatively, we can leverage a certain quantity in calculus to find all input variables that the output neuron $Z^{(L)}_{:,n_L}$ depends on. Denote $c^{n_L}_m$ to be this dependency quantity between the input variable $X_{:,m}$ and the output variable $Z^{(L)}_{:,n_L}$.
Suppose we assume that $c^{n_L}_m \neq 0$ if $Z^{(L)}_{:,n_L}$ indeed depends on $X_{:,m}$. *Show* what the formula of $c^{n_L}_m$ is and *outline* how to deduce the receptive field size from a collection $\{c^{n_L}_m\}^T_{m=1}$.
Hints:

1. The quantity of our interest is related to one of the norms we looked at in Exercise 4.
2. One might also consider the n_1-th spatial value of $Z^{(1)}$ from the example in Figure 2; the value can be viewed as the output of a linear layer with a weight matrix containing only three non-zero weights. Observe that the number of non-zero weights is the same as the receptive field size of the n_1-th spatial value of $Z^{(1)}$.

Solution:
The quantity is based on the partial derivatives

$$c^{n_L}_m = \sum_{i=1}^{d} \frac{\partial}{\partial X_{i,m}} \left(\sum_{j=1}^{d_L} Z^{(L)}_{j,n_L} \right).$$

In the case of input with one spatial dimension, we can deduce the size of the the receptive field via

$$\texttt{RF}(Z^{(L)}_{:,n_L}) = \sum_{m=1}^{T} [|c^{n_l}_m| > 0].$$

(d) This question is the programming part of the exercise. We will implement the solution of (c) and use the implementation to calculate the receptive fields of neurons in various layers of VGG16. Download the programming files on ISIS and follow the instructions.

Exercise 3: Translation Equivalence of Convolution (Bonus: 10+10 P)

Let $A \in \mathbb{R}^{d \times T}$ to be a feature map with T spatial locations and $\delta \in \{1, \ldots, T-1\}$. Denote $\mathcal{T}_\delta : \mathbb{R}^{d \times T} \to \mathbb{R}^{d \times T}$ to be a δ-unit translation along the spatial dimension, i.e.

$$A \mapsto \mathcal{T}_\delta(A) = \hat{A}$$

and $\forall m \in \{0, \ldots, T-1\}$[3]

$$(\mathcal{T}_\delta(A))_m = A_{(m-\delta) \bmod T}.$$

[3]For this excercise, we note that the index m start with zero.

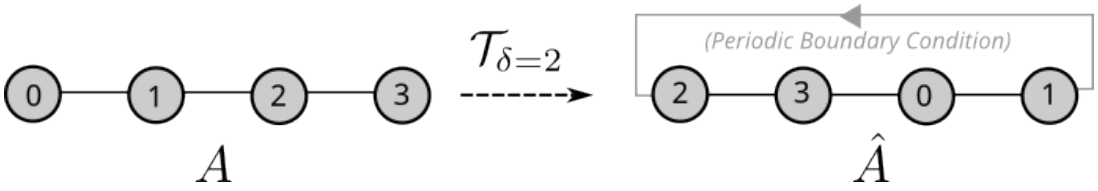

Figure 5: Translating the feature map A by δ with the periodic boundary condition.

The symbol $a \bmod b$ represents the modulo operator, and the $(\cdot) \bmod T$ is known as the periodic boundary condition (PBC). We illustrate the translation $\mathcal{T}_\delta$ and PBC in Figure 5.
Suppose $W \in \mathbb{R}^{1\times d\times k}$ is a convolution weight with kernel size $k = 1$ and output dimension 1. Under PBC, we write the convolution operator of W on the feature map A as

$$(A \star W)_m = \sum_{\tau=1}^{k}\sum_{j=1}^{d} A_{j,(m+\tau-1) \bmod T} W_{j,\tau}.$$

In the following, you will show that the convolution operator of W on $T_\delta(A)$ satisfies the translation equivalence property

$$\forall m : (\mathcal{T}_\delta(A \star W))_m = (\mathcal{T}_\delta(A) \star W)_m. \tag{3}$$

We first start by expanding $(T_\delta(A) \star W)_m$ in terms of A and W

$$\begin{aligned}(\mathcal{T}_\delta(A) \star W)_m &= \sum_{\tau=1}^{k}\sum_{j=1}^{d} \mathcal{T}_\delta(A)_{j,(m+\tau-1) \bmod T} W_{j,\tau} \\ &= \sum_{\tau}\sum_{j} A_{j,[((m+\tau-1) \bmod T)-\delta] \bmod T} W_{j,\tau}\end{aligned}$$

(a) *Show* that

1. $[((m+\tau-1) \bmod T) - \delta] \bmod T = (m+\tau-1-\delta) \bmod T$
2. $(m+\tau-1-\delta) \bmod T = (m' + (\tau-1)) \bmod T$ where $m' := (m-\delta) \bmod T$

Hints: For the modulo operator, we have the following identities

1. $(a+b) \bmod c = [(a \bmod c) + (b \bmod c)] \bmod c$
2. $(a \bmod T) \bmod T = a \bmod T$

Solution:
(Part 1) We first define $\alpha := (m+\tau-1) \bmod T$ and $\delta' := -\delta$. Then, we have

$$\begin{aligned}[((m+\tau-1) \bmod T) - \delta] \bmod T &= (\alpha + \delta') \bmod T \\ &= [(\alpha \bmod T) + (\delta' \bmod T)] \bmod T && \text{(Hint 1: } \rightarrow\text{)} \\ &= [\alpha + (\delta' \bmod T)] \bmod T && \text{(Hint 2)} \\ &= [((m+\tau-1) \bmod T) + (\delta' \bmod T)] \bmod T && \text{(Definition of } \alpha\text{)} \\ &= (m+\tau-1+\delta') \bmod T && \text{(Hint 1: } \leftarrow\text{)} \\ &= (m+\tau-1-\delta) \bmod T && \text{(Definition of } \delta'\text{)}\end{aligned}$$

(Part 2)

$$\begin{aligned}(m+\tau-1-\delta) \bmod T &= [(m-\delta) + (\tau-1)] \bmod T \\ &= [(m-\delta) \bmod T + (\tau-1) \bmod T] \bmod T && \text{(Hint 1: } \rightarrow\text{)} \\ &= [m' + (\tau-1) \bmod T] \bmod T && \text{(Definition of } m'\text{)} \\ &= [m' \bmod T + ((\tau-1) \bmod T) \bmod T] \bmod T && \text{(Hint 1: } \rightarrow\text{)} \\ &= [m' \bmod T + (\tau-1) \bmod T] \bmod T && \text{(Hint 2)} \\ &= (m' + (\tau-1)) \bmod T && \text{(Hint 1: } \leftarrow\text{)}\end{aligned}$$

(b) Using the results of (a), *Show* that Eq. 3 holds.
Solution:

$$
\begin{aligned}
(\mathcal{T}_\delta(A) \star W)_m &= \sum_{\tau=1}^{k} \sum_{j=1}^{d} A_{j,[((m+\tau-1) \bmod T)-\delta] \bmod T} W_{j,\tau} \\
&= \sum_{\tau=1}^{k} \sum_{j=1}^{d} A_{j,(m+\tau-1-\delta) \bmod T} W_{j,\tau} && \text{(from (a.1))} \\
&= \sum_{\tau=1}^{k} \sum_{j=1}^{d} A_{j,(m'+\tau-1) \bmod T} W_{j,\tau} && \text{(from (a.2))} \\
&= (A \star W)_{m'} && \text{(Definition of Convolution under PBC)} \\
&= (A \star W)_{(m-\delta) \bmod T} && \text{(Definition of } m' \text{ in (a.2))} \\
&= \big(\mathcal{T}_\delta(A \star W)\big)_m && \text{(Definition of } \mathcal{T}_\delta\text{)}
\end{aligned}
$$