PROBLEMS ON CNN

Dr Divya Meena Sundaram

Sr. Assistant Prof. Grade 2

SCOPE

VIT-AP University

Given the following 5 × 5 input matrix X:

$$X = \begin{bmatrix} 3 & 2 & 1 & 5 & 4 \\ 4 & 1 & 3 & 2 & 0 \\ 2 & 5 & 4 & 1 & 3 \\ 0 & 2 & 1 & 4 & 5 \\ 1 & 3 & 2 & 0 & 4 \end{bmatrix}$$

Layer 1: Convolution Layer (2×2 filter, stride=1, no padding)

W = [[0, 1], [-1, 0]]

Layer 2: ReLU Activation

Layer 3: Max Pooling Layer (2×2 filter, stride=2, no padding)

Layer 4: Fully Connected Layer with Softmax (3-class)

W_fc = [[0.2, -0.3, 0.4, 0.1],
[-0.1, 0.5, -0.2, 0.3],
[0.3, -0.4, 0.2, -0.5]] **# 3x4 matrix**

b = [0.05, -0.1, 0.2] **# 1x3 bias vector**

Compute the **softmax probabilities** after applying all layers step by step

Step 1: Input Matrix X

$$X = \begin{bmatrix} 3 & 2 & 1 & 5 & 4 \\ 4 & 1 & 3 & 2 & 0 \\ 2 & 5 & 4 & 1 & 3 \\ 0 & 2 & 1 & 4 & 5 \\ 1 & 3 & 2 & 0 & 4 \end{bmatrix}$$

Step 2: Convolution Layer

- Stride = 1
- No Padding
- Output Size: 4×4

$$W = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Output size: Since the input is 5×5 and the filter is 2×2, the output feature map will be:

$$\text{Output size} = (5 - 2)/1 + 1 = 4$$

So the output feature map will be of size 4×4.

Convolution Output: $\begin{bmatrix} -2 & 0 & 2 & 2 \\ -1 & -2 & -2 & -1 \\ 5 & 2 & 0 & -1 \\ 1 & -2 & 2 & 5 \end{bmatrix}$

Row 1 Computation:

1. **Position (0,0)**

$$\begin{bmatrix} 3 & 2 \\ 4 & 1 \end{bmatrix}$$

$$(3 \times 0) + (2 \times 1) + (4 \times -1) + (1 \times 0) = 0 + 2 - 4 + 0 = -2$$

2. **Position (0,1)**

$$\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$$

$$(2 \times 0) + (1 \times 1) + (1 \times -1) + (3 \times 0) = 0 + 1 - 1 + 0 = 0$$

3. **Position (0,2)**

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

$$(1 \times 0) + (5 \times 1) + (3 \times -1) + (2 \times 0) = 0 + 5 - 3 + 0 = 2$$

4. **Position (0,3)**

$$\begin{bmatrix} 5 & 4 \\ 2 & 0 \end{bmatrix}$$

$$(5 \times 0) + (4 \times 1) + (2 \times -1) + (0 \times 0) = 0 + 4 - 2 + 0 = 2$$

$$W = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$X = \begin{bmatrix} 3 & 2 & 1 & 5 & 4 \\ 4 & 1 & 3 & 2 & 0 \\ 2 & 5 & 4 & 1 & 3 \\ 0 & 2 & 1 & 4 & 5 \\ 1 & 3 & 2 & 0 & 4 \end{bmatrix}$$

Row 2 Computation:

5. **Position (1,0)**

$$\begin{bmatrix} 4 & 1 \\ 2 & 5 \end{bmatrix}$$

$$(4 \times 0) + (1 \times 1) + (2 \times -1) + (5 \times 0) = 0 + 1 - 2 + 0 = -1$$

6. **Position (1,1)**

$$\begin{bmatrix} 1 & 3 \\ 5 & 4 \end{bmatrix}$$

$$(1 \times 0) + (3 \times 1) + (5 \times -1) + (4 \times 0) = 0 + 3 - 5 + 0 = -2$$

7. **Position (1,2)**

$$\begin{bmatrix} 3 & 2 \\ 4 & 1 \end{bmatrix}$$

$$(3 \times 0) + (2 \times 1) + (4 \times -1) + (1 \times 0) = 0 + 2 - 4 + 0 = -2$$

8. **Position (1,3)**

$$\begin{bmatrix} 2 & 0 \\ 1 & 3 \end{bmatrix}$$

$$(2 \times 0) + (0 \times 1) + (1 \times -1) + (3 \times 0) = 0 + 0 - 1 + 0 = -1$$

$$X = \begin{bmatrix} 3 & 2 & 1 & 5 & 4 \\ 4 & 1 & 3 & 2 & 0 \\ 2 & 5 & 4 & 1 & 3 \\ 0 & 2 & 1 & 4 & 5 \\ 1 & 3 & 2 & 0 & 4 \end{bmatrix}$$

Row 3 Computation:

9. **Position (2,0)**

$$\begin{bmatrix} 2 & 5 \\ 0 & 2 \end{bmatrix}$$

$$(2 \times 0) + (5 \times 1) + (0 \times -1) + (2 \times 0) = 0 + 5 - 0 + 0 = 5$$

10. **Position (2,1)**

$$\begin{bmatrix} 5 & 4 \\ 2 & 1 \end{bmatrix}$$

$$(5 \times 0) + (4 \times 1) + (2 \times -1) + (1 \times 0) = 0 + 4 - 2 + 0 = 2$$

$$X = \begin{bmatrix} 3 & 2 & 1 & 5 & 4 \\ 4 & 1 & 3 & 2 & 0 \\ 2 & 5 & 4 & 1 & 3 \\ 0 & 2 & 1 & 4 & 5 \\ 1 & 3 & 2 & 0 & 4 \end{bmatrix}$$

11. **Position (2,2)**

$$\begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix}$$

$$(4 \times 0) + (1 \times 1) + (1 \times -1) + (4 \times 0) = 0 + 1 - 1 + 0 = 0$$

12. **Position (2,3)**

$$\begin{bmatrix} 1 & 3 \\ 4 & 5 \end{bmatrix}$$

$$(1 \times 0) + (3 \times 1) + (4 \times -1) + (5 \times 0) = 0 + 3 - 4 + 0 = -1$$

Row 4 Computation:

13. Position (3,0)

$$\begin{bmatrix} 0 & 2 \\ 1 & 3 \end{bmatrix}$$

$$(0 \times 0) + (2 \times 1) + (1 \times -1) + (3 \times 0) = 0 + 2 - 1 + 0 = 1$$

14. Position (3,1)

$$\begin{bmatrix} 2 & 1 \\ 3 & 2 \end{bmatrix}$$

$$(2 \times 0) + (1 \times 1) + (3 \times -1) + (2 \times 0) = 0 + 1 - 3 + 0 = -2$$

15. Position (3,2)

$$\begin{bmatrix} 1 & 4 \\ 2 & 0 \end{bmatrix}$$

$$(1 \times 0) + (4 \times 1) + (2 \times -1) + (0 \times 0) = 0 + 4 - 2 + 0 = 2$$

16. Position (3,3)

$$\begin{bmatrix} 4 & 5 \\ 0 & 4 \end{bmatrix}$$

$$(4 \times 0) + (5 \times 1) + (0 \times -1) + (4 \times 0) = 0 + 5 - 0 + 0 = 5$$

$$X = \begin{bmatrix} 3 & 2 & 1 & 5 & 4 \\ 4 & 1 & 3 & 2 & 0 \\ 2 & 5 & 4 & 1 & 3 \\ 0 & 2 & 1 & 4 & 5 \\ 1 & 3 & 2 & 0 & 4 \end{bmatrix}$$

Convolution Output:

$$\begin{bmatrix} -2 & 0 & 2 & 2 \\ -1 & -2 & -2 & -1 \\ 5 & 2 & 0 & -1 \\ 1 & -2 & 2 & 5 \end{bmatrix}$$

Step 3: ReLU Activation

Apply ReLU: $f(x) = \max(0, x)$

Convolution Output:

$$\begin{bmatrix} -2 & 0 & 2 & 2 \\ -1 & -2 & -2 & -1 \\ 5 & 2 & 0 & -1 \\ 1 & -2 & 2 & 5 \end{bmatrix} \longrightarrow$$

ReLU Output:

$$\begin{bmatrix} 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 \\ 5 & 2 & 0 & 0 \\ 1 & 0 & 2 & 5 \end{bmatrix}$$

Step 4: Max Pooling Layer

- Filter size: 2×2
- Stride: 2
- No Padding
- Output Size: 2×2

$\longrightarrow$

Max Pooling Output:

$$\begin{bmatrix} 0 & 2 \\ 5 & 5 \end{bmatrix}$$

Step 5: Flattening

Convert the 2×2 matrix into a 1D vector: $\begin{bmatrix} 0 & 2 & 5 & 5 \end{bmatrix}$

Step 6: Fully Connected Layer

Weights W_{FC} (size 3×4):

$$W_{FC} = \begin{bmatrix} 0.2 & -0.3 & 0.4 & 0.1 \\ -0.1 & 0.5 & -0.2 & 0.3 \\ 0.3 & -0.4 & 0.2 & -0.5 \end{bmatrix}$$

Bias b (size 1×3):

$$b = \begin{bmatrix} 0.05 & -0.1 & 0.2 \end{bmatrix}$$

$$Z = W_{FC} \cdot X + b$$

$$Z = \begin{bmatrix} 1.95 & 1.4 & -2.1 \end{bmatrix}$$

Step 7: Softmax Function

$$\text{Softmax}(Z_i) = \frac{e^{Z_i}}{\sum e^{Z_j}}$$

$$\begin{bmatrix} 0.6272 & 0.3619 & 0.0109 \end{bmatrix}$$

Answer

Matrix Multiplication $W_{FC} \cdot X$

Step 6 (cont.)

$$\begin{bmatrix} 0.2 & -0.3 & 0.4 & 0.1 \\ -0.1 & 0.5 & -0.2 & 0.3 \\ 0.3 & -0.4 & 0.2 & -0.5 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 2 \\ 5 \\ 5 \end{bmatrix}$$

1. **First row:**

$$(0.2 \times 0) + (-0.3 \times 2) + (0.4 \times 5) + (0.1 \times 5)$$

$$= 0 - 0.6 + 2 + 0.5 = 1.9$$

2. **Second row:**

$$(-0.1 \times 0) + (0.5 \times 2) + (-0.2 \times 5) + (0.3 \times 5)$$

$$= 0 + 1 - 1 + 1.5 = 1.5$$

3. **Third row:**

$$(0.3 \times 0) + (-0.4 \times 2) + (0.2 \times 5) + (-0.5 \times 5)$$

$$= 0 - 0.8 + 1 - 2.5 = -2.3$$

Add Bias b

$$Z = \begin{bmatrix} 1.9 \\ 1.5 \\ -2.3 \end{bmatrix} + \begin{bmatrix} 0.05 \\ -0.1 \\ 0.2 \end{bmatrix}$$

$$Z = \begin{bmatrix} 1.95 \\ 1.4 \\ -2.1 \end{bmatrix}$$

CALCULATING PARAMETERES WITHOUT BATCH NORMALIZATION

```python
import tensorflow as tf
from tensorflow.keras import layers, models

model = models.Sequential([
    layers.Conv2D(filters=32, kernel_size=(3,3), strides=1, padding="same", activation='relu', input_shape=(64, 64, 3)),
    layers.MaxPooling2D(pool_size=(2,2), strides=2, padding="valid"),
    layers.Conv2D(filters=64, kernel_size=(3,3), strides=1, padding="same", activation='relu'),
    layers.MaxPooling2D(pool_size=(2,2), strides=2, padding="valid"),
    layers.Flatten(),
    layers.Dense(units=128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(units=2, activation='softmax')
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 64, 64, 32)	896
max_pooling2d (MaxPooling2D)	(None, 32, 32, 32)	0
conv2d_1 (Conv2D)	(None, 32, 32, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 64)	0
flatten (Flatten)	(None, 16384)	0
dense (Dense)	(None, 128)	2,097,280
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 2)	258

Total params: 2,116,930 (8.08 MB)

Trainable params: 2,116,930 (8.08 MB)

Non-trainable params: 0 (0.00 B)

Calculating total no: of parameters

```
5       layers.Conv2D(filters=32, kernel_size=(3,3), strides=1, padding="same", activation='relu', input_shape=(64, 64, 3)),
```

Step 1: First Conv2D Layer

- **Input Shape**: (64, 64, 3)
- **Filter Size**: (3×3)
- **Strides**: 1 (moves 1 pixel at a time)
- **Padding**: `"same"` (keeps output size same as input)
- **Number of Filters**: 32

$$\text{Output Size} = \frac{\text{Input Size} - \text{Filter Size} + 2 \times \text{Padding}}{\text{Stride}} + 1$$

Since **padding="same"**, output size remains **(64, 64, 32)**.

Parameter Calculation

Each filter has:

$$\text{Weights} = \text{Filter Size} \times \text{Filter Size} \times \text{Input Channels} + \text{Bias}$$

$$= (3 \times 3 \times 3) + 1 = 28$$

Since there are **32 filters**, total parameters:

$$32 \times 28 = 896$$

```python
6    layers.MaxPooling2D(pool_size=(2,2), strides=2, padding="valid"),
```

Step 2: First MaxPooling2D Layer

- **Input size: (64, 64, 32)**
- Pool Size: (2×2)
- Strides: 2
- Padding: `"valid"` (no padding)

$$\text{Output Size} = \frac{\text{Input Size} - \text{Pool Size}}{\text{Stride}} + 1$$

$$\frac{64 - 2}{2} + 1 = 32$$

New Output Shape: (32, 32, 32)

Parameters: 0 (Pooling has no trainable weights)

```python
    layers.Conv2D(filters=64, kernel_size=(3,3), strides=1, padding="same", activation='relu'),
```

Step 3: Second Conv2D Layer

- **Input Shape**: (32, 32, 32)
- **Filter Size**: (3×3)
- **Filters**: 64
- **Padding**: `"same"` (keeps size same)

New Output Shape: (32, 32, 64) (same as input due to "same" padding)

Parameter Calculation

Each filter has:

$$\text{Weights} = (3 \times 3 \times 32) + 1 = 289$$

Since there are **64 filters**, total parameters:

$$64 \times 289 = 18,496$$

```
    layers.MaxPooling2D(pool_size=(2,2), strides=2, padding="valid"),
```

Step 4: Second MaxPooling2D Layer

- **Input size: (32, 32, 64)**
- Pool Size: (2×2)
- Strides: 2

$$\frac{32-2}{2}+1=16$$

New Output Shape: (16, 16, 64)

Parameters: 0

```
    layers.Flatten(),
```

Step 5: Flatten Layer

- Input Shape: (16, 16, 64)
- Output Shape: (16 × 16 × 64) = (16384)
- Parameters: 0 (just reshaping)

```
    layers.Dense(units=128, activation='relu'),
```

Step 6: Fully Connected Dense Layer

- Input Shape: (16384)
- Neurons: 128

New Output Shape: (128)

$$\text{Weights} = \text{Input Features} \times \text{Neurons}$$

$$= 16384 \times 128 = 2,097,152$$

Each neuron has a bias term, so the number of **bias parameters** is:

$$\text{Bias} = \text{Neurons} = 128$$

$$\text{Total Parameters} = \text{Weights} + \text{Bias}$$

$$= 2,097,152 + 128 = 2,097,280$$

```
    layers.Dropout(0.5),
```

Step 7: Dropout Layer

- No change in shape
- Output Shape: (128)
- Parameters: 0

```
12        layers.Dense(units=2, activation='softmax')
```

Step 8: Output Dense Layer

- Input Shape: (128)
- Output Neurons: 2

$$\text{Weights} = 128 \times 2 = 256$$

$$\text{Bias} = 2$$

$$\text{Total Parameters} = 258$$

Final Output Shape: (2) (Softmax gives two probability scores)

1. **Conv2D layers keep the spatial dimensions the same** (if `padding="same"`) or reduce them (if `padding="valid"`).
2. **Pooling layers reduce the spatial dimensions** by **half.**
3. **Flatten converts feature maps into a 1D vector.**
4. **Dense layers fully connect neurons** for classification.
5. **Softmax outputs class probabilities.**

Layer	Type	Output Shape	Parameters
Conv2D	32 filters, 3×3 kernel	(64, 64, 32)	896
MaxPooling2D	2×2 pooling	(32, 32, 32)	0
Conv2D	64 filters, 3×3 kernel	(32, 32, 64)	18,496
MaxPooling2D	2×2 pooling	(16, 16, 64)	0
Flatten	Convert to 1D vector	(16384)	0
Dense (FC)	128 neurons	(128)	2,097,280
Dropout	50% dropout	(128)	0
Dense (Output)	2 neurons (Softmax)	(2)	258

Total Trainable Parameters = 2,116,930

```
Total params: 2,116,930 (8.08 MB)
Trainable params: 2,116,930 (8.08 MB)
Non-trainable params: 0 (0.00 B)
```

CALCULATING PARAMETERES WITH BATCH NORMALIZATION

```python
import tensorflow as tf
from tensorflow.keras import layers, models

model = models.Sequential([
    layers.Conv2D(filters=32, kernel_size=(3,3), strides=1, padding="same", activation=None, input_shape=(64, 64, 3))
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPooling2D(pool_size=(2,2), strides=2, padding="valid"),

    layers.Conv2D(filters=64, kernel_size=(3,3), strides=1, padding="same", activation=None),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPooling2D(pool_size=(2,2), strides=2, padding="valid"),

    layers.Flatten(),
    layers.Dense(units=128, activation=None),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.Dropout(0.5),

    layers.Dense(units=2, activation='softmax')
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

1. First Conv2D Layer

- Input shape: (64, 64, 3)
- Filters: 32
- Kernel size: (3,3)
- Weights: $(3 \times 3 \times 3) \times 32 = 864$
- Biases: 32
- Total: $864 + 32 = 896$

2. Batch Normalization (First)

- BN has **4 parameters per filter** (scale, shift, mean, variance).
- Total BN params: $4 \times 32 = 128$

3. Second Conv2D Layer

- Input shape: (32, 32, 32)
- Filters: 64
- Kernel size: (3,3)
- Weights: $(3 \times 3 \times 32) \times 64 = 18,432$
- Biases: 64
- Total: $18,432 + 64 = 18,496$

4. Batch Normalization (Second)

- Total BN params: $4 \times 64 = 256$

5. Dense Layer (128 Neurons)

- **Input size**: Flattened output size from conv layers:
  - Input to dense: $15 \times 15 \times 64 = 14,400$
- **Weights**: $14,400 \times 128 = 1,843,200$
- **Biases**: 128
- **Total**: $1,843,200 + 128 = 1,843,328$

6. Batch Normalization (Third)

- **Total BN params**: $4 \times 128 = 512$

7. Final Dense Layer (Output Layer)

- **Weights**: $128 \times 2 = 256$
- **Biases**: 2
- **Total**: $256 + 2 = 258$

Final Parameter Count

Layer	Weights	Bias	BN Params	Total Params
Conv2D (32 filters)	864	32	128	**1,024**
Conv2D (64 filters)	18,432	64	256	**18,752**
Dense (128 units)	1,843,200	128	512	**1,843,840**
Dense (2 units)	256	2	0	**258**
Total	**1,862,752**	**226**	**896**	**1,863,874**

Thus, the final model has **1,863,874 parameters**.