NN IO Shape Calculation

Deep Learning
EE 298/CoE 197/EE 197/ECE 197
University of the Philippines Diliman
2022

Conv2d

Check conv2d info here.

Shape:

- Input: $(N, C_{in}, H_{in}, W_{in})$ or (C_{in}, H_{in}, W_{in})
- Output: $(N, C_{out}, H_{out}, W_{out})$ or $(C_{out}, H_{out}, W_{out})$, where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

MaxPool2d

Check maxpool2d info here.

Shape:

- Input: (N, C, H_{in}, W_{in}) or (C, H_{in}, W_{in})
- Output: (N, C, H_{out}, W_{out}) or (C, H_{out}, W_{out}), where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 * \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 * \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

CNN Shape Calculation

```python
class CNN(pl.LightningModule):
    def __init__(self, num_channel, num_class):
        super().__init__()
        self.num_channel = num_channel
        self.num_class = num_class

        self.conv1 = nn.Conv2d(num_channel, 32, kernel_size=3)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3)
        self.conv3 = nn.Conv2d(64, 128, kernel_size=3)
        self.pool = nn.MaxPool2d(2,2)
        self.relu = nn.ReLU()
        self.fc1 = nn.Linear(128*4*4, 128)
        self.fc2 = nn.Linear(128, num_class)
        self.criterion = nn.CrossEntropyLoss()

    def forward(self, x):
        x = self.pool(self.relu(self.conv1(x)))
        x = self.pool(self.relu(self.conv2(x)))
        x = self.relu(self.conv3(x))
        x = x.view(x.shape[0], -1)
        x = self.relu(self.fc1(x))
        return self.fc2(x)
```

Demo link here.

Input: [B, 3, 32, 32] (cifar10)

CNN Shape Calculation

```python
class CNN(pl.LightningModule):
    def __init__(self, num_channel, num_class):
        super().__init__()
        self.num_channel = num_channel
        self.num_class = num_class

        self.conv1 = nn.Conv2d(num_channel, 32, kernel_size=3)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3)
        self.conv3 = nn.Conv2d(64, 128, kernel_size=3)
        self.pool = nn.MaxPool2d(2,2)
        self.relu = nn.ReLU()
        self.fc1 = nn.Linear(128*4*4, 128)
        self.fc2 = nn.Linear(128, num_class)
        self.criterion = nn.CrossEntropyLoss()

    def forward(self, x):
        x = self.pool(self.relu(self.conv1(x)))
        x = self.pool(self.relu(self.conv2(x)))
        x = self.relu(self.conv3(x))
        x = x.view(x.shape[0], -1)
        x = self.relu(self.fc1(x))
        return self.fc2(x)
```

Demo link here.

Input: [B, 3, 32, 32] (cifar10)

After conv1:

$$C_{out} = 32, H_{out} = W_{out} = \frac{32+2*0-1*(3-1)-1}{1} + 1 = 30$$

After pool1:

$$C_{out} = 32, H_{out} = W_{out} = \frac{30+2*0-1*(2-1)-1}{2} + 1 = 15$$

Final: [B, 32, 15, 15]

CNN Shape Calculation

```python
class CNN(pl.LightningModule):
    def __init__(self, num_channel, num_class):
        super().__init__()
        self.num_channel = num_channel
        self.num_class = num_class

        self.conv1 = nn.Conv2d(num_channel, 32, kernel_size=3)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3)
        self.conv3 = nn.Conv2d(64, 128, kernel_size=3)
        self.pool = nn.MaxPool2d(2,2)
        self.relu = nn.ReLU()
        self.fc1 = nn.Linear(128*4*4, 128)
        self.fc2 = nn.Linear(128, num_class)
        self.criterion = nn.CrossEntropyLoss()

    def forward(self, x):
        x = self.pool(self.relu(self.conv1(x)))
        x = self.pool(self.relu(self.conv2(x)))
        x = self.relu(self.conv3(x))
        x = x.view(x.shape[0], -1)
        x = self.relu(self.fc1(x))
        return self.fc2(x)
```

Demo link here.

Input: [B, 32, 15, 15]

After conv2:

$$C_{out} = 64, H_{out} = W_{out} = \frac{15+2*0-1*(3-1)-1}{1} + 1 = 13$$

After pool2:

$$C_{out} = 64, H_{out} = W_{out} = \frac{13+2*0-1*(2-1)-1}{2} + 1 = 6$$

Final: [B, 64, 6, 6]

CNN Shape Calculation

```python
class CNN(pl.LightningModule):
    def __init__(self, num_channel, num_class):
        super().__init__()
        self.num_channel = num_channel
        self.num_class = num_class

        self.conv1 = nn.Conv2d(num_channel, 32, kernel_size=3)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3)
        self.conv3 = nn.Conv2d(64, 128, kernel_size=3)
        self.pool = nn.MaxPool2d(2,2)
        self.relu = nn.ReLU()
        self.fc1 = nn.Linear(128*4*4, 128)
        self.fc2 = nn.Linear(128, num_class)
        self.criterion = nn.CrossEntropyLoss()

    def forward(self, x):
        x = self.pool(self.relu(self.conv1(x)))
        x = self.pool(self.relu(self.conv2(x)))
        x = self.relu(self.conv3(x))
        x = x.view(x.shape[0], -1)
        x = self.relu(self.fc1(x))
        return self.fc2(x)
```

Demo link here.

Input: [B, 64, 6, 6]

After conv3:

$$C_{out} = 128, H_{out} = W_{out} = \frac{6+2*0-1*(3-1)-1}{1} + 1 = 4$$

CNN Shape Calculation

```python
class CNN(pl.LightningModule):
    def __init__(self, num_channel, num_class):
        super().__init__()
        self.num_channel = num_channel
        self.num_class = num_class

        self.conv1 = nn.Conv2d(num_channel, 32, kernel_size=3)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3)
        self.conv3 = nn.Conv2d(64, 128, kernel_size=3)
        self.pool = nn.MaxPool2d(2,2)
        self.relu = nn.ReLU()
        self.fc1 = nn.Linear(128*4*4, 128)
        self.fc2 = nn.Linear(128, num_class)
        self.criterion = nn.CrossEntropyLoss()

    def forward(self, x):
        x = self.pool(self.relu(self.conv1(x)))
        x = self.pool(self.relu(self.conv2(x)))
        x = self.relu(self.conv3(x))
        x = x.view(x.shape[0], -1)
        x = self.relu(self.fc1(x))
        return self.fc2(x)
```

Demo link here.

Input: [B, 64, 6, 6]

After conv3:

$$C_{out} = 128, H_{out} = W_{out} = \frac{6+2*0-1*(3-1)-1}{1} + 1 = 4$$

Our linear layer:

$$H_{in} = C \times H \times W = 128 * 4 * 4 = 2048$$