Fire Module

Learn about the central component of SqueezeNet, the fire module.

Chapter Goals:

- Learn strategies for decreasing the number of parameters in a model
- Understand how the fire module works and why it's effective
- Write your own fire module function

A. Decreasing parameters

In order to make a smaller model, we need to decrease the number of weights per convolution layer. There are three ways to decrease the number of weights in a convolution layer:

- Decrease the kernel size
- Decrease the number of filters used
- Decrease the number of input channels

We don't necessarily want to reduce the number of filters we use, since using a variety of filters allows us to extract different hidden features from the input. However, there are ways to decrease the kernel size and number of input channels while still maintaining good model performance.

B. Kernel size

The size of a kernel represents the amount of spatial information it can capture. For example, a 1x1 kernel will only capture the channel information for individual pixels, while a 3x3 kernel will aggregate the information between adjacent pixels within each 3x3 square of the input data.

Although larger kernels can capture more information, it comes at the cost of additional parameters. A convolution layer that uses 3x3 kernels will use 9x as many parameters as a layer that uses 1x1 kernels. A good strategy for balancing performance and parameter count is to use a mix of larger and smaller size kernels.

C. Intermediate layer

The way we decrease the number of input channels is by adding an intermediate convolution layer. Though this may seem counter-intuitive, since adding an extra layer introduces additional kernel weights, it can drastically decrease the number of parameters used in a layer. Consider a convolution layer with 100 filters and 3x3 kernels. Given the input has 50 channels, we can use the equation from chapter 1 to calculate the number of parameters in the layer:

$$P = 3 \times 3 \times 100 \times 50 + 100 = 45{,}100$$

Now, consider the case where we first apply an intermediate convolution layer with 10 filters and 1x1 kernels. The intermediate output will have 10 channels. The number of parameters used in the intermediate layer, P_I, is

$$P_I = 1 \times 1 \times 10 \times 50 + 10 = 510$$

Then if we pass the intermediate output into our original convolution layer, the total number of parameters used becomes

$$P = P_I + 3 \times 3 \times 100 \times 10 + 100 = 9{,}610$$

By adding the intermediate layer we decrease the number of parameters by a factor of nearly 5.

D. Fire module

The fire module, the key building block of SqueezeNet, incorporates the ideas from the previous two sections. It uses an intermediate convolution layer, referred to as a *squeeze layer*, then passes the intermediate output into an *expand layer* with a larger number of filters.

The expand layer contains two convolution layers with an equal number of filters. One of the layers uses 1x1 kernels, while the other uses 3x3 kernels. The mix of 1x1 kernels decreases the number of parameters used. The outputs of the two layers have the same size, since both layers use the same number of filters.

The two outputs are then concatenated along the channel dimension (doubling the number of channels) to produce the overall output of the fire

module.

To concatenate, we use the function `tf.concat`, which takes in two required arguments:

- `values`: A list of tensors to concatenate.
- `axis`: The dimension to concatenate along.

Since the channel dimension is the last dimension, we use `-1` as the `axis`.

```
import tensorflow as tf

print(expand1x1.shape)
print(expand3x3.shape)
output = tf.concat([expand1x1, expand3x3], axis=-1)
print(output.shape)
```

Example: Concatenation of expand1x1 and expand3x3 in the fire module.

The ratio of the number of filters in the squeeze layer vs. the expand layer is known as the *squeeze ratio*. A larger squeeze ratio (i.e. increasing the number of filters in the squeeze layer) can improve the model performance up to a certain extent, at the cost of increased parameter count.

Below is the full code for a fire module:

```
import tensorflow as tf

class SqueezeNetModel(object):
    # __init__ and other functions omitted

    # Convolution layer wrapper
    def custom_conv2d(self, inputs, filters, kernel_size, name):
        return tf.layers.conv2d(
            inputs=inputs,
            filters=filters,
            kernel_size=kernel_size,
            activation=tf.nn.relu,
            padding='same',
            name=name)

    # SqueezeNet fire module
    def fire_module(self, inputs, squeeze_depth, expand_depth, name):
        with tf.variable_scope(name):
            squeezed_inputs = self.custom_conv2d(
                inputs,
                squeeze_depth,
```

```
            [1, 1],
            'squeeze')
        expand1x1 = self.custom_conv2d(
            squeezed_inputs,
            expand_depth,
            [1, 1],
            'expand1x1')
        expand3x3 = self.custom_conv2d(
            squeezed_inputs,
            expand_depth,
            [3, 3],
            'expand3x3')
        return tf.concat([expand1x1, expand3x3], axis=-1)
```

Fire module and conv layer wrapper within the SqueezeNetModel.