Consider this model:

```
class Net(nn.Module):
        def __init__(self):
            super(Net, self).__init__()
            self.fc1 = nn.Linear(10,5, bias=True)
            self.fc2 = nn.Linear(5, 2, bias=True)
        def forward(self, x):
            x = self.fc1(x)
            x = nn.ReLU()(x)
            x = self.fc2(x)
            return x
```

What is the number of trainable parameters in this model

(A) Depends on the input

(B) 60

(C) 62

(D) **[Ans]** 67

(E) None of these

Consider this model:

```
class Net(nn.Module):
        def __init__(self):
            super(Net, self).__init__()
            self.fc1 = nn.Linear(10,5, bias=True)
            self.fc2 = nn.Linear(5, 2, bias=True)
        def forward(self, x):
            x = self.fc1(x)
            x = nn.ReLU()(x)
            x = self.fc2(x)
            return x
```

Now consider that a tensor of shape $(5, 10)$ is given as input to this model. What is the shape of the output tensor?

(A) $(10, 2)$

(B) **[Ans]** $(5, 2)$

(C) $(5, 5)$

(D) Cannot be answered with given information

(E) None of these

Consider this model:

```
class Net(nn.Module):
        def __init__(self):
            super(Net, self).__init__()
            self.fc1 = nn.Linear(10,5, bias=True)
            self.fc2 = nn.Linear(5, 2, bias=True)
        def forward(self, x):
            x = self.fc1(x)
            y = nn.ReLU()(x)
            z = self.fc2(y)
            return z
```

Now consider that a tensor of shape $(5, 10)$ is given as input to this model. What is the shape of the tensor y in the forward function ?

(A) $(10, 2)$

(B) $(5, 2)$

(C) **[Ans]** $(5, 5)$

(D) Cannot be answered with given information

(E) None of these

Which of these are valid ways to address the problem of overfitting?

(A) **[Ans]** Data augmentation

(B) **[Ans]** L1 regularization of weights

(C) Increase the number of layers in model

(D) **[Ans]** Decrease the number of layers in model

(E) **[Ans]** Add dropout to one of the layers

(F) None of these

Consider this model with just 1 convolutional layer.

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv = nn.Conv2d(in_channels=3,out_channels=64,
                              kernel_size=(5,5), bias=False)
    def forward(self, x):
        x = self.conv(x)
        return x
```

Consider that we want to input a tensor of shape (10,3,512,512) to this model. Then

(A) The number of trainable parameters is $64 \times 5 \times 5$

(B) **[Ans]** The number of trainable parameters is $64 \times 3 \times 5 \times 5$

(C) The shape of output tensor is $(10, 3, 508, 508)$

(D) **[Ans]** The shape of output tensor is $(10, 64, 508, 508)$

(E) None of these