Assignment 5

MLP Group

1. Environment setup (Same as last assignment):
    1. Download Anaconda (What is Anaconda?)on your computer - link to ARM version with Python 3.12
    2. Copy the anaconda installer to the Jetson nano/Raspberry PI via scp
    3. Install Anaconda. It is highly recommended to install via terminal by just executing the downloaded file.
        - During installation you can decide if you want to have anaconda at startup - I would recommend to don't do that to avoid system breaks
    4. Download pytorch installer from your computer at this url: http://download.pytorch.org/whl/cpu/torch/. You have to download the version 2.2.2 for the python version installed in your device and for the right architecture (arm64). Then transfer it to the Jetson nano/Raspberry PI
    5. Download torchvision installer from your computer at this url: http://download.pytorch.org/whl/cpu/torch/. You have to download the version 0.17 for the python version installed in your device and for the right architecture (arm64). Then transfer it to the Jetson nano/Raspberry PI
2. Download the parameters from Studon the parameters under Assignment 5/model.pth
   This is the model code

```
class ConvNet(nn.Module):
 def __init__(self):
     super(ConvNet, self).__init__()
     self.conv1 = nn.Conv2d(1, 32, 3)
     self.pool = nn.MaxPool2d(2, 2)
     self.conv2 = nn.Conv2d(32, 64, 3)
     self.fc1 = nn.Linear(64 * 5 * 5, 128)
     self.fc2 = nn.Linear(128, 10)
     self.relu = nn.ReLU()
```

```python
def forward(self, x):
    x = self.pool(self.relu(self.conv1(x)))
    x = self.pool(self.relu(self.conv2(x)))
    x = x.view(-1, 64 * 5 * 5)
    x = self.relu(self.fc1(x))
    x = self.fc2(x)
    return x
```

3. Implement or re-use from the last assignment the same api at endpoint `'/'`, where given the API request with input data the *image* and the *true label* you perform the prediction with the model with loaded weights and store the pairs (image, label)
4. Store Top-1 and Top-5 classification accuracy
5. Once you perform 1000 predictions, starts to finetuning procedure where you continue the training of the model on the embedded device using *X_train.pth* and *y_train.pth*
6. After the finetuning, store the model in a file with name *model_finetuned.pth*
7. Test the model on the test set with both version of the model i.e *model.pth* and *model_pretrained.pth*