Putting Everything Together:

```
device = torch.device("cuda"
                      if torch.cuda.is_available()
                      else
                      "cpu")
model = models.resnet50(pretrained = True)
```

freeze:

```
for param in model.parameters():
    param.requires_grad = False
```

replace the pre-trained classifier:

```
classifier = nn.Sequential(nn.Linear(2048, 512),
                           nn.ReLU(),
                           nn.Dropout(p=0.2),
                           nn.Linear(512, 2),
                           nn.LogSoftmax(dim=1))
model.fc = classifier
```

```
criterion = nn.NLLLoss()
optimizer = optim.Adam(model.fc.parameters(),
                       lr=0.003)
```

move the model to the available device:

```
model.to(device)
```

set hyper params

```
epochs = 1
steps = 0
running_loss = 0
print_every = 5
```

start actual training →

```
for epoch in range(epochs):
    for ims, labs in trainloader:
        steps += 1
```

move to the available device ←

```
ims, labs = ims.to(device), labs.to(device)
```

```
optimizer.zero_grad()
logps = model(images)
loss = criterion(logps, labels)
loss.backward()
optimizer.step()

running_loss += loss.item()
```

frequency for validation →

```
if steps % print_every == 0:
    model.eval()
    test_loss = 0
    accuracy = 0
```

remember to transfer to gpu:

```
ims, labs = ims.to(device)
            labs.to(device)
```

Validation

```
for ims, labs in testloader:
    logps = model(ims)
    loss = criterion(logps, labs)
    test_loss += loss.item()

    ps = torch.exp(logps)
    top_ps, top_class = ps.topk(1,
                                dim=1)
    equality = top_class ==
               labels.view(*top_cls.shape)
    accuracy += torch.mean(
        equality.type(torch.FloatTensor
        )).item()
```

```
running_loss = 0
model.train()
```