Exercício 1:

np.random.seed(0)
torch.manual_seed(0)
random.seed(0)

	Modelo 1	Modelo 2	Modelo 3	Modelo 4
epochs	30	30	30	30
batch size	32	32	32	32
learning rate	0.001	0.001	0.001	0.001
size splits	Treino:47999 Validação:12000 Teste:9999	Treino:47999 Validação:12000 Teste:9999	Treino:47999 Validação:12000 Teste:9999	Treino:47999 Validação:12000 Teste:9999
layers + activation functions	CNNModel_1((layer1): Sequential((0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1)) (1): ReLU() (2): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation=1, ceil_mode=False)) (layer2): Sequential((0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1)) (1): ReLU() (2): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation=1, ceil_mode=False)) (fc1): Linear(in_features=800, out_features=100, bias=True) (act1): ReLU() (fc2): Linear(in_features=100, out_features=10, bias=True) (act2): Softmax(dim=1))	CNNModel_2((layer1): Sequential((0): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1)) (1): ReLU() (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)) (layer2): Sequential((0): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1)) (1): ReLU() (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)) (fc1): Linear(in_features=800, out_features=10, bias=True))	CNNModel_3((layer1): Sequential((0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (2): ReLU() (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)) (layer2): Sequential((0): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1)) (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (2): ReLU() (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)) (fc1): Linear(in_features=2304, out_features=600, bias=True) (drop): Dropout(p=0.25, inplace=False) (fc2): Linear(in_features=600, out_features=120, bias=True) (fc3): Linear(in_features=120, out_features=10, bias=True))	CNNModel_4((layer1): Sequential((0): Conv2d(1, 32, kernel_size=(5, 5), stride=(1, 1)) (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (2): ReLU() (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False) (4): Dropout2d(p=0.2, inplace=False)) (fc1): Linear(in_features=4608, out_features=128, bias=True) (fc2): Linear(in_features=128, out_features=10, bias=True))
loss function	CrossEntropyLoss()	CrossEntropyLoss()	CrossEntropyLoss()	CrossEntropyLoss()
optimization function	SGD(model.parameters(), lr=LEARNING_RATE)	SGD(model.parameters(), lr=LEARNING_RATE)	SGD(model.parameters(), lr=LEARNING_RATE)	SGD(model.parameters(), lr=LEARNING_RATE)
accuracy	0.849	0.970	0.987	0.980

Exercício 2:

np.random.seed(0)
torch.manual_seed(0)
random.seed(0)

	Modelo 1	Modelo 2	Modelo 3	Modelo 4
epochs	50	50	15	50
batch size	32	32	32	32
learning rate	0.001	0.001	0.001	0.001
size splits	Treino:48000 Validação:12000 Teste:10000	Treino:48000 Validação:12000 Teste:10000	Treino:48000 Validação:12000 Teste:10000	Treino:48000 Validação:12000 Teste:10000
layers + activation functions	CNNModel_1((layer1): Sequential((0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1)) (1): ReLU() (2): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation=1, ceil_mode=False)) (layer2): Sequential((0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1)) (1): ReLU() (2): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation=1, ceil_mode=False)) (fc1): Linear(in_features=800, out_features=100, bias=True) (act1): ReLU() (fc2): Linear(in_features=100, out_features=10, bias=True) (act2): Softmax(dim=1))	CNNModel_2((layer1): Sequential((0): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1)) (1): ReLU() (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)) (layer2): Sequential((0): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1)) (1): ReLU() (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)) (fc1): Linear(in_features=800, out_features=10, bias=True))	CNNModel_3((layer1): Sequential((0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (2): ReLU() (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)) (layer2): Sequential((0): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1)) (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (2): ReLU() (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)) (fc1): Linear(in_features=2304, out_features=600, bias=True) (drop): Dropout(p=0.25, inplace=False) (fc2): Linear(in_features=600, out_features=120, bias=True) (fc3): Linear(in_features=120, out_features=10, bias=True))	CNNModel_4((layer1): Sequential((0): Conv2d(1, 32, kernel_size=(5, 5), stride=(1, 1)) (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (2): ReLU() (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False) (4): Dropout2d(p=0.2, inplace=False)) (fc1): Linear(in_features=4608, out_features=128, bias=True) (fc2): Linear(in_features=128, out_features=10, bias=True))
loss function	CrossEntropyLoss()	CrossEntropyLoss()	CrossEntropyLoss()	CrossEntropyLoss()
optimization function	SGD(model.parameters(), lr=LEARNING_RATE)	SGD(model.parameters(), lr=LEARNING_RATE)	SGD(model.parameters(), lr=LEARNING_RATE)	SGD(model.parameters(), lr=LEARNING_RATE)
accuracy	0.794	0.854	0.902	0.907