1.logistic

No hidden

model = LogisticClassifier(input\_dim=data\_all[0].shape[1], hidden\_dim=None, reg=0.2)

solver = Solver(model, data,

update\_rule='sgd\_momentum',

optim\_config={

'learning\_rate': 1e-0,

},

lr\_decay=0.95,

num\_epochs=20, batch\_size=50,

print\_every=100)

solver.train() (Epoch 50 / 50) train acc: 0.964000; val\_acc: 0.952000

With hidden

model = LogisticClassifier(input\_dim=data\_all[0].shape[1], hidden\_dim=32, reg=0.3)

solver = Solver(model, data,

update\_rule='sgd\_momentum',

optim\_config={

'learning\_rate': 8e-2,

},

lr\_decay=0.9,

num\_epochs=20, batch\_size=50,

print\_every=100)

solver.train()

(Epoch 50 / 50) train acc: 0.962000; val\_acc: 0.952000

2. SVM

No hidden

model = SVM(input\_dim=data\_all[0].shape[1], hidden\_dim=None, reg=0.1)

solver = Solver(model, data,

update\_rule='sgd\_momentum',

optim\_config={

'learning\_rate': 1e-0,

},

lr\_decay=0.95,

num\_epochs=50, batch\_size=50,

print\_every=100)

solver.train()

(Epoch 50 / 50) train acc: 0.964000; val\_acc: 0.944000

With hidden

model = SVM(input\_dim=data\_all[0].shape[1], hidden\_dim=32, reg=0.1)

solver = Solver(model, data,

update\_rule='sgd\_momentum',

optim\_config={

'learning\_rate': 1e-1,

},

lr\_decay=0.95,

num\_epochs=50, batch\_size=50,

print\_every=100)

solver.train()

(Epoch 50 / 50) train acc: 0.968000; val\_acc: 0.928000

3. Softmax

No hidden

With hidden

model = SoftmaxClassifier(hidden\_dim=64, reg=0.001)

solver = Solver(model, data,

update\_rule='adam',

optim\_config={

'learning\_rate': 1e-3,

},

lr\_decay=0.95,

num\_epochs=20, batch\_size=50,

print\_every=100)

solver.train()

(Epoch 20 / 20) train acc: 0.994000; val\_acc: 0.967400

3. VGG

nn.Conv2d(3, 16, 3, padding=1),

nn.BatchNorm2d(16),

nn.ReLU(),

nn.MaxPool2d(2),

max-pooling layer, size 2

nn.Conv2d(16, 32, 3, padding=1),

nn.BatchNorm2d(32),

nn.ReLU(),

nn.MaxPool2d(2),

# Stage 3

# TODO: convolutional layer, input channels 16, output channels 32, filter size 3

# TODO: convolutional layer, input channels 32, output channels 32, filter size 3

# TODO: max-pooling layer, size 2

nn.Conv2d(32, 32, 3, padding=1),

nn.BatchNorm2d(32),

nn.ReLU(),

nn.Conv2d(32, 32, 3, padding=1),

nn.BatchNorm2d(32),

nn.ReLU(),

nn.MaxPool2d(2),

# Stage 4

# TODO: convolutional layer, input channels 32, output channels 64, filter size 3

# TODO: convolutional layer, input channels 64, output channels 64, filter size 3

# TODO: max-pooling layer, size 2

nn.Conv2d(32, 64, 3, padding=1),

nn.BatchNorm2d(64),

nn.ReLU(),

nn.Conv2d(64, 64, 3, padding=1),

nn.BatchNorm2d(64),

nn.ReLU(),

nn.MaxPool2d(2),

# Stage 5

# TODO: convolutional layer, input channels 64, output channels 64, filter size 3

# TODO: convolutional layer, input channels 64, output channels 64, filter size 3

# TODO: max-pooling layer, size 2

nn.Conv2d(64, 64, 3, padding=1),

nn.BatchNorm2d(64),

nn.ReLU(),

nn.Conv2d(64, 64, 3, padding=1),

nn.BatchNorm2d(64),

nn.ReLU(),

nn.MaxPool2d(2)

batch\_size=100

lr=0.001

74 %