Chapter 14 Report:
Classifying Images with CNNs

Quan Pham

June 18, 2025

Contents

1 Abstract

In this chapter, I studied convolutional neural networks (CNNs) and their application in image classification tasks. CNNs are designed to extract hierarchical features from images, which makes them highly effective for vision-related problems.

2 Key Concepts

2.1 Convolutional Neural Networks

- CNNs mimic the human visual cortex and automatically learn features from raw data.
- Feature hierarchies: low-level features $\rightarrow$ high-level representations.
- Key ideas: **sparse connectivity** and **parameter sharing**.
- Pooling layers reduce spatial dimensions and help with generalization.

2.2 Discrete Convolution

$$y[i] = \sum_{k=-\infty}^{+\infty} x[i-k] \cdot w[k]$$
$$Y[i,j] = \sum_{k_1} \sum_{k_2} X[i-k_1, j-k_2] \cdot W[k_1, k_2]$$

Important hyperparameters: padding (full, same, valid), stride.

2.3 Pooling Layers

- Max-pooling and mean-pooling.
- Improve robustness to noise, reduce overfitting.
- Can use overlapping or non-overlapping pooling.

2.4 Convolution layer

- Handling multiple input/output channels:

$$Z^{\text{conv}}[:,:,k] = \sum_{c=1}^{C_{in}} W[:,:,c,k] * X[:,:,c]$$
$$Z[:,:,k] = Z^{\text{conv}} + b[k]$$
$$A[:,:,k] = \sigma(Z[:,:,k])$$

2.5 Regularization

- L2 regularization (weight decay) and dropout help prevent overfitting.
- Dropout encourages robust feature learning.

2.6 Loss Functions

- Binary classification: `BCEWithLogitsLoss`, `BCELoss`
- Multiclass: `CrossEntropyLoss`, `NLLLoss`

2.7 Other Techniques

- **Data augmentation**: enhances generalization.
- **Global average pooling**: reduces parameters.

3 Algorithms and Models Studied

4 Code Examples and Practical Exercises

4.1 Implementing a deep CNN using PyTorch

Prepare the MNIST dataset:

```python
import torchvision
from torchvision import transforms
import torch
x = torch.tensor([5, 5, 5])

from torch import nn

image_path = '../data/'
transform = transforms.Compose([
    transforms.ToTensor()
])
mnist_dataset = torchvision.datasets.MNIST(root=image_path, train=True, transform=transform, download=False)

from torch.utils.data import Subset
mnist_valid_dataset = Subset(mnist_dataset, torch.arange(10000))
mnist_train_datset = Subset(mnist_dataset, torch.arange(10000, len(mnist_dataset)))
mnist_test_dataset = torchvision.datasets.MNIST(root=image_path, train=False, transform=transform, download=False)
```

```

from torch.utils.data import DataLoader
batch_size = 64
torch.manual_seed(1)
train_dl = DataLoader(mnist_train_datset, batch_size, shuffle=True)
valid_dl = DataLoader(mnist_valid_dataset, batch_size, shuffle=False)
```

Construct the CNN model using the `Sequential` class:

```
model = nn.Sequential()
model.add_module(
    'conv1',
    nn.Conv2d(
        in_channels=1, out_channels=32, kernel_size=5, padding=2
    )
)
model.add_module('relu1', nn.ReLU())
model.add_module('pool1', nn.MaxPool2d(kernel_size=2))
model.add_module(
    'conv2',
    nn.Conv2d(
        in_channels=32, out_channels=64, kernel_size=5, padding=2
    )
)
model.add_module('relu2', nn.ReLU())
model.add_module('pool2', nn.MaxPool2d(kernel_size=2))
model.add_module('fc1', nn.Linear(3136, 1024))
model.add_module('relu3', nn.ReLU())
model.add_module('dropout', nn.Dropout(p=0.5))
model.add_module('fc2', nn.Linear(1024, 10))
```

Train accuracy and validation accuracy for 20 epochs:

```
Epoch 1 accuracy: 0.9496 val_accuracy: 0.9792
...
Epoch 10 accuracy: 0.9964 val_accuracy: 0.9902
...
Epoch 20 accuracy: 0.9981 val_accuracy: 0.9909
```

4.2 Smile classification from face images using CNN

```
get_smile = lambda attr: attr[18]

transform_train = transforms.Compose([
    transforms.RandomCrop([178, 178]),
```

```
    transforms.RandomHorizontalFlip(),
    transforms.Resize([64, 64]),
    transforms.ToTensor(),
])

transform = transforms.Compose([
    transforms.CenterCrop([178, 178]),
    transforms.Resize([64, 64]),
    transforms.ToTensor(),
])
```

```
celeba_train_dataset = torchvision.datasets.CelebA(image_path, split='train', target_type='attr', download=False, transform=transform_train, target_transform=get_smile)
celeba_train_dataset = Subset(celeba_train_dataset, torch.arange(16000))
celeba_valid_dataset = Subset(celeba_valid_dataset, torch.arange(1000))
```

```
batch_size = 32
torch.manual_seed(1)
train_dl = DataLoader(celeba_train_dataset, batch_size, shuffle=True)
valid_dl = DataLoader(celeba_valid_dataset, batch_size, shuffle=False)
test_dl = DataLoader(celeba_test_dataset, batch_size, shuffle=False)
```

```
model = nn.Sequential()
model.add_module('conv1', nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, padding=1))
model.add_module('relu1', nn.ReLU())
model.add_module('pool1', nn.MaxPool2d(kernel_size=2))

model.add_module('conv2', nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, padding=1))
model.add_module('relu2', nn.ReLU())
model.add_module('pool2', nn.MaxPool2d(kernel_size=2))

model.add_module('conv3', nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, padding=1))
model.add_module('relu3', nn.ReLU())
model.add_module('pool3', nn.MaxPool2d(kernel_size=2))

model.add_module('conv4', nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3, padding=1))
model.add_module('relu4', nn.ReLU())

model.add_module('pool4', nn.AvgPool2d(kernel_size=8))
```

```
model.add_module('flatten', nn.Flatten())
model.add_module('fc', nn.Linear(256, 1))
model.add_module('sigmoid', nn.Sigmoid())
```

```
loss_fn = nn.BCELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

```
Epoch 1 accuracy: 0.6350 val_accuracy: 0.7100
...
Epoch 15 accuracy: 0.8899 val_accuracy: 0.8990
...
Epoch 30 accuracy: 0.9139 val_accuracy: 0.9030
```

5 Learnings and Challenges

5.1 Key Takeaways

5.2 Challenges Encountered

5.3 Opening Questions

6 Conclusion and Furture work

6.1 Conclusion

6.2 Future work

References

[1] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.