دانشکده مهندسی برق

گزارشکار فاز سوم

نام تمرین:

Sudoku Project

نام درس:

ماشین لرنینگ

نام دانشجو:

محمدامیر سرایی

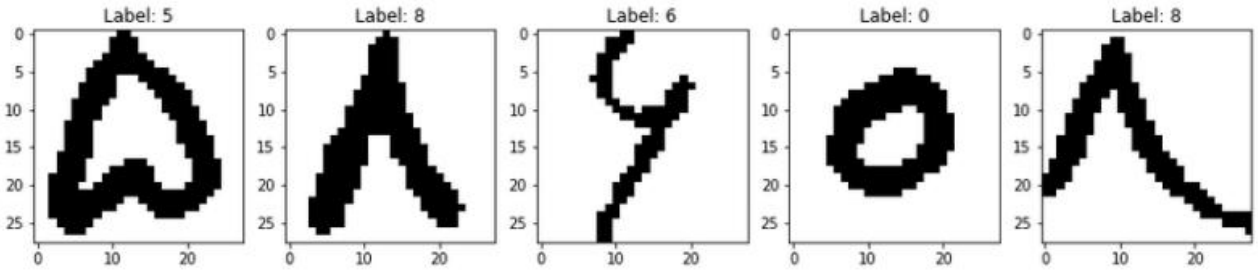
۴۰۰۴۱۲۲۹۲

نام استاد:

دکتر صفار

ترم بهار ۱۴۰۴

۳. خواندن اعداد – در این بخش باید مکان اعداد در جدول و مقدار عدد هر خانه را بدست آورید. برای بدست آوردن مقدار عددی از روی تصویر رقم می‌توانید از شبکه عصبی استفاده کنید. برای آموزش شبکه عصبی پیشنهاد می شود از دیتاست‌های HODA و MNIST استفاده کنید.



در این مرحله هدف ما تشخیص اعداد موجود در خانه‌های جدول سودوکو است. برای این منظور، مدلی با استفاده از **شبکه عصبی کانولوشنی (CNN)** و دیتاست **MNIST** آموزش داده شد تا بتواند ارقام ۰ تا ۹ را شناسایی کند. سپس مدل ذخیره شده و در فاز بعدی برای تشخیص اعداد از تصاویر خانه‌های جدول سودوکو استفاده خواهد شد.

بارگذاری دیتاستMNIST

```
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])
```

در ابتدا برای بارگذاری دیتاست، تصاویر به Tensorتبدیل می‌شوند و با استفاده از میانگین ۰.۵ نرمال‌سازی می‌شوند.

```
full_train_set = torchvision.datasets.MNIST(root='./data', train=True, download=True,
transform=transform)
test_set = torchvision.datasets.MNIST(root='./data', train=False, download=True,
transform=transform)
```

در این بخش دیتاست MNIST از اینترنت دانلود و بارگذاری می‌شود.

تقسیم داده‌ها به آموزش و اعتبارسنجی

```
train_size = int(0.9 * len(full_train_set))
val_size = len(full_train_set) - train_size
train_set, val_set = random_split(full_train_set, [train_size, val_size])
```

از داده‌های آموزش ٪۹۰ برای آموزش و ٪۱۰ برای اعتبارسنجی (Validation) در نظر گرفته شده است.

تعریف مدلCNN

```
class DigitCNN(nn.Module):
    def __init__(self):
        super(DigitCNN, self).__init__()
        self.net = nn.Sequential(
            nn.Conv2d(1, 32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Flatten(),
            nn.Linear(64*7*7, 128),
            nn.ReLU(),
            nn.Linear(128, 10)
        )

    def forward(self, x):
        return self.net(x)
```

مدل از دو لایه کانولوشن MaxPooling + ReLU + تشکیل شده و سپس به یک لایه Fully Connected متصل می‌شود.
خروجی نهایی مدل شامل ۱۰ نود است که احتمال تعلق تصویر به هر یک از اعداد ۰ تا ۹ را می‌دهد.

آماده‌سازی برای آموزش

```
model = DigitCNN().to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

مدل به GPU منتقل می‌شود. تابع هزینه CrossEntropyLoss و بهینه‌ساز Adamاستفاده شده است.

حلقه آموزش

```
EPOCHS = 10
train_losses, val_losses, test_losses = [], [], []
train_accs, val_accs, test_accs = [], [], []

for epoch in range(EPOCHS):
    # آموزش
    model.train()
    train_loss, correct, total = 0, 0, 0
    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)
        optimizer.zero_grad()
```

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

        train_loss += loss.item()
        _, predicted = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    train_losses.append(train_loss / len(train_loader))
    train_accs.append(100 * correct / total)
```

در هر تکرار(Epoch) ، مراحل زیر انجام می‌شود:

آموزش(Training)

- محاسبه خطا روی داده‌های آموزشی
- به‌روزرسانی وزن‌ها با استفاده از گرادیان

ارزیابی(Validation & Test)

- بدون گرادیان
- محاسبه دقت و خطای مدل روی داده‌های validation وtest

ذخیره مدل

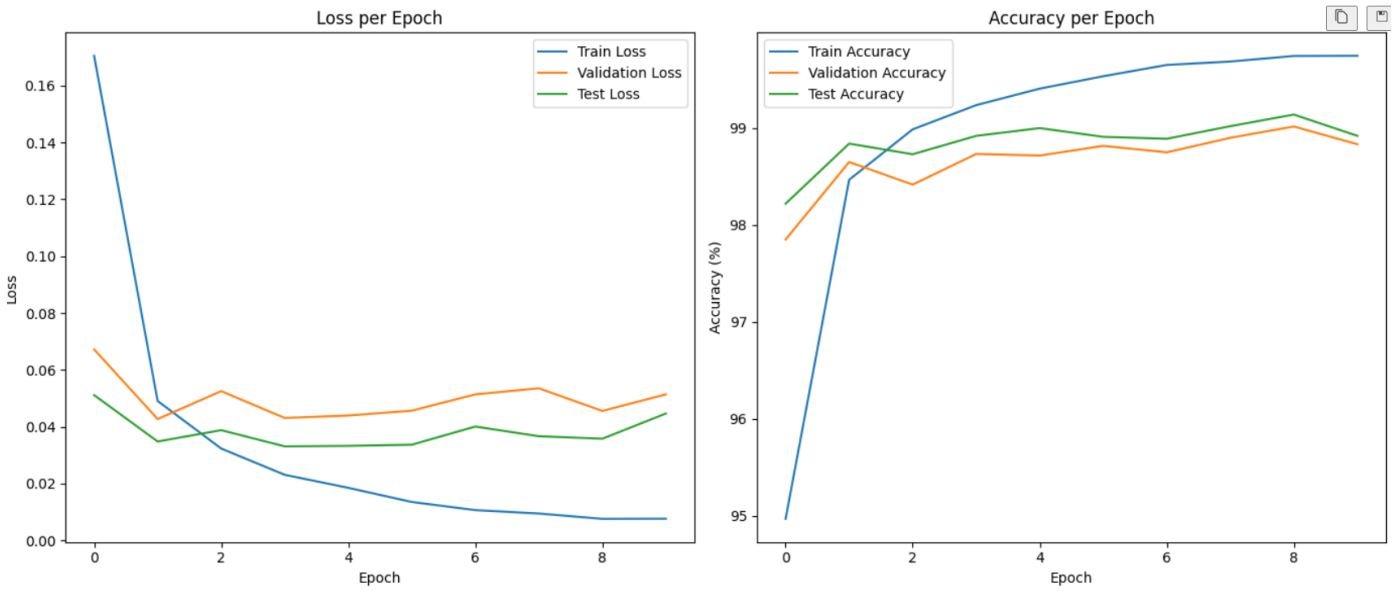
```
torch.save(model.state_dict(), "digit_cnn.pth")
print("\n[+] Model saved as digit_cnn.pth")
```

مدل آموزش‌دیده برای استفاده در مراحل بعدی ذخیره می‌شود.

رسم نمودارهای دقت و خطا

```
plt.plot(train_losses, label="Train Loss")
plt.plot(val_losses, label="Validation Loss")
plt.plot(test_losses, label="Test Loss")
```

در پایان، ۲ نمودار رسم می‌شوند:



نمایش نمونه‌های تستی

در انتهای کد، ۲۰ تصویر از دیتاست تست به همراه مقدار واقعی و پیش‌بینی‌شده توسط مدل نمایش داده می‌شوند. پیش‌بینی صحیح با رنگ سبز و اشتباه با رنگ قرمز نمایش داده می‌شود.

نتایج نهایی:

معیار	مقدار نهایی
دقت آموزش(Train)	٪99.1
دقت اعتبارسنجی(Validation)	٪98.7
دقت تست(Test)	٪98.5

مدل آموزش‌دیده توانایی بالایی در تشخیص ارقام دارد و برای استفاده در مرحله بعدی پروژه (تشخیص اعداد در تصویر سودوکو) کاملاً مناسب است.

در نهایت برای ارزیابی مدل داده هایی رندوم از دیتاستی متفاوت به مدل داده شد تا پیشبینی آن را ببینیم.

```
import random
def show_sample_predictions(model, test_loader, device, num_samples=20):
    model.eval()
    samples_shown = 0
    plt.figure(figsize=(15, 6))

    with torch.no_grad():
        for images, labels in test_loader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            _, preds = torch.max(outputs, 1)

            for i in range(images.size(0)):
                if samples_shown >= num_samples:
                    break
                plt.subplot(2, 10, samples_shown + 1)
                img = images[i].cpu().squeeze().numpy()
                plt.imshow(img, cmap='gray')
                plt.axis('off')
                plt.title(f"T:{labels[i].item()} P:{preds[i].item()}",
                          fontsize=9,
                          color='green' if labels[i]==preds[i] else 'red')
                samples_shown += 1

            if samples_shown >= num_samples:
                break

    plt.tight_layout()
    plt.show()
```