Bài tập số 6: CNN cho MNIST

1. **Chuẩn bị dữ liệu:**

```
num_classes = 10
input_shape = (28, 28, 1)

# Load the data and split it between train and test sets
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()

# Scale images to the [0, 1] range
x_train = x_train.astype("float32") / 255
x_test = x_test.astype("float32") / 255
# Make sure images have shape (28, 28, 1)
x_train = np.expand_dims(x_train, -1)
x_test = np.expand_dims(x_test, -1)
print("x_train shape:", x_train.shape)
print(x_train.shape[0], "train samples")
print(x_test.shape[0], "test samples")


# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

- Bộ dữ liệu MNIST được tải lên và được chuẩn hóa về khoảng [0, 1] bằng cách chia cho 255.
- Dữ liệu ảnh được reshape để thêm kênh chiều cuối cùng (kích thước đầu vào là 28x28x1).
- Nhãn được chuyển thành ma trận nhị phân (one-hot encoding) bằng hàm `to_categorical`.

2. **Xây dựng mô hình:**

```
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)

model.summary()
```

- **Đầu vào**: Dữ liệu có kích thước 28x28 với 1 kênh (ảnh đen trắng).
- **Hai lớp tích chập (Convolutional layers)**: Mỗi lớp sử dụng bộ lọc có kích thước 3x3 và hàm kích hoạt ReLU để tạo ra các đặc trưng quan trọng từ ảnh.
- **Hai lớp pooling (MaxPooling layers)**: Giảm kích thước của dữ liệu đầu ra từ lớp tích chập, giúp giảm số lượng tham số và tăng hiệu suất tính toán.
- **Lớp Dropout**: Giảm thiểu hiện tượng quá khớp (overfitting) bằng cách ngẫu nhiên loại bỏ các nơ-ron trong quá trình huấn luyện.
- **Lớp Dense với Softmax**: Dùng để phân loại ảnh thành 10 lớp (các chữ số từ 0 đến 9).

3. **Huấn luyện mô hình:**

```
batch_size = 128
epochs = 15

model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])

model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validation_split=0.1)
```

- Sử dụng `adam` làm tối ưu hóa và `categorical_crossentropy` làm hàm mất mát, cùng với các chỉ số độ chính xác.
- Mô hình sẽ được huấn luyện trong 15 epochs với kích thước batch là 128, và 10% dữ liệu huấn luyện sẽ được sử dụng làm tập xác thực.

4. **Đánh giá:**

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

- Sau khi huấn luyện, mô hình được đánh giá trên tập kiểm tra, và hiển thị giá trị mất mát và độ chính xác cuối cùng.

```
Test loss: 0.025680875405669212
Test accuracy: 0.991100013256073
```