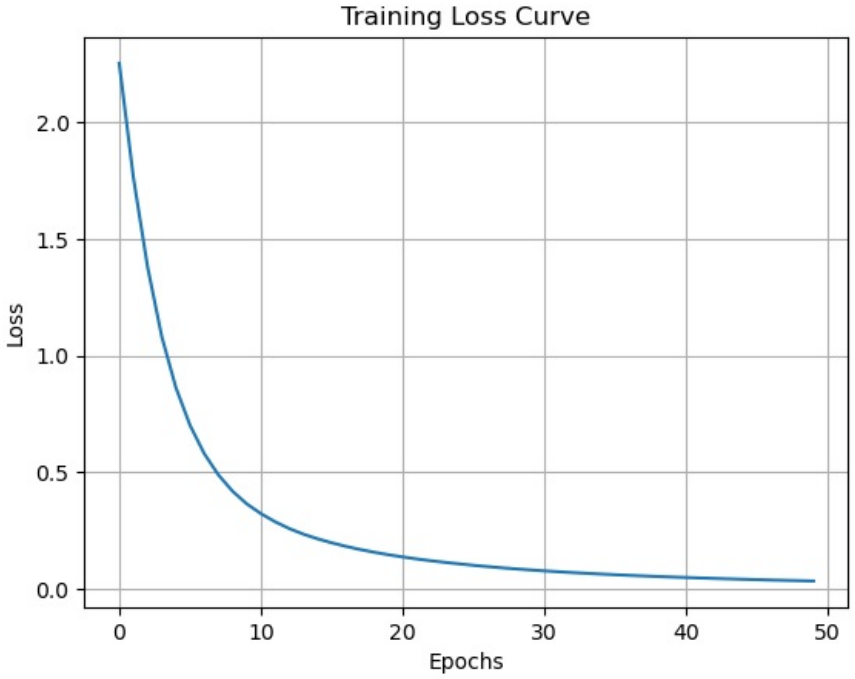
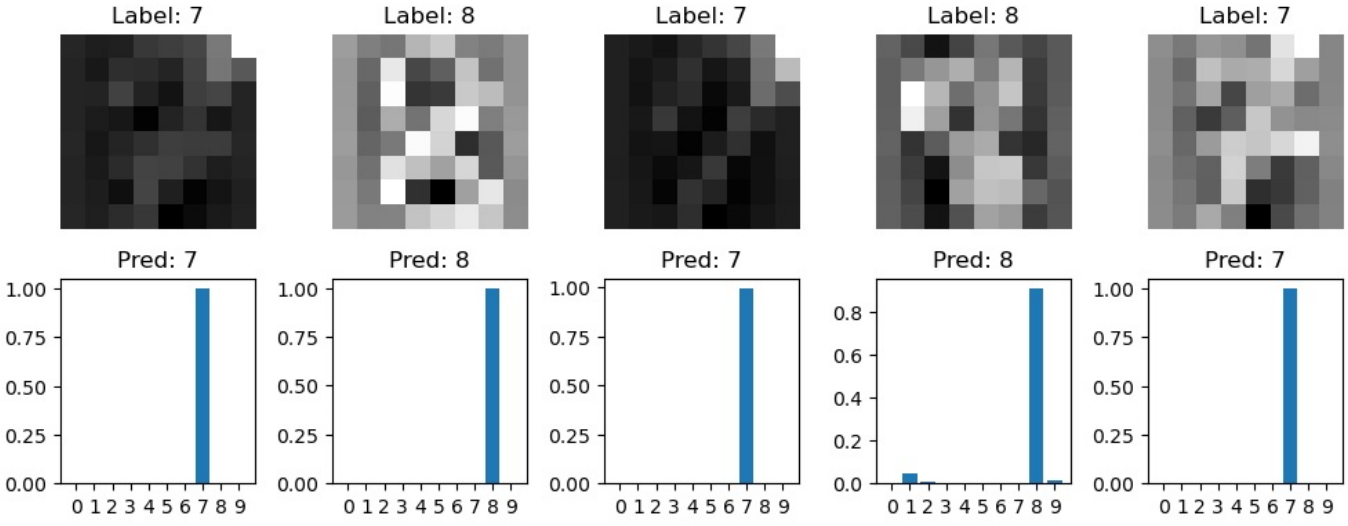
In [1]:

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, accuracy_score

# Load inbuilt Digits dataset
digits = load_digits()
X = digits.data  # Each image is 8x8 pixels, flattened to 64 features
y = digits.target

# Split into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
X_test = scaler.transform(X_test)

# Train MLP model
mlp = MLPClassifier(hidden_layer_sizes=(128,), activation='relu', solver='adam', max_iter=50, random_state=42)
mlp.fit(X_train, y_train)

# Evaluate model
val_preds = mlp.predict(X_val)
test_preds = mlp.predict(X_test)

print("Validation Accuracy:", accuracy_score(y_val, val_preds))
print("Test Accuracy:", accuracy_score(y_test, test_preds))
print("\nClassification Report:\n", classification_report(y_test, test_preds))

# Plot training loss curve
plt.plot(mlp.loss_curve_)
plt.title("Training Loss Curve")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.grid(True)
plt.show()

# Display sample predictions
def display_predictions(images, labels, model):
    plt.figure(figsize=(10, 4))
    for i in range(5):
        plt.subplot(2, 5, i + 1)
        plt.imshow(images[i].reshape(8, 8), cmap='gray')
        plt.title(f"Label: {labels[i]}")
        plt.axis('off')

        plt.subplot(2, 5, i + 6)
        probs = model.predict_proba(images[i].reshape(1, -1))[0]
        plt.bar(range(10), probs)
        plt.title(f"Pred: {np.argmax(probs)}")
        plt.xticks(range(10))
    plt.tight_layout()
    plt.show()

# Show predictions on random test samples
indices = np.random.choice(len(X_test), 5, replace=False)
display_predictions(X_test[indices], y_test[indices], mlp)
```

```
Validation Accuracy: 0.9740740740740741
Test Accuracy: 0.9740740740740741

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.97      0.98        31
           1       1.00      0.96      0.98        24
           2       0.93      1.00      0.96        27
           3       0.97      0.97      0.97        33
           4       0.94      1.00      0.97        29
           5       0.96      0.93      0.94        27
           6       1.00      0.96      0.98        23
           7       1.00      1.00      1.00        27
           8       1.00      1.00      1.00        25
           9       0.96      0.96      0.96        24

    accuracy                           0.97       270
   macro avg       0.98      0.97      0.97       270
weighted avg       0.97      0.97      0.97       270
```

```
C:\Users\User\anaconda3\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:691: ConvergenceWarni
ng: Stochastic Optimizer: Maximum iterations (50) reached and the optimization hasn't converged yet.
  warnings.warn(
```





In [ ]: