Exercise 4: Convolutional Neural Networks (CNNs)

Objective:

Apply the concepts of Convolutional Neural Networks (CNNs) as explained in the provided notebook to a new dataset. You will build, train, and evaluate a CNN model on one of the datasets listed below.

Instructions:

1. **Choose a Dataset:** Select one of the following datasets to work with:
    a. **Fashion-MNIST:** A dataset of Zalando's article images—consisting of 70,000 grayscale images in 10 categories.
    b. **Street View House Numbers (SVHN):** Real-world images for digit recognition.
    c. **Intel Image Classification:** Images categorized into six classes of natural scenes.
    d. **Tiny ImageNet:** A subset of the ImageNet dataset with 200 classes.
2. **Data Preparation:**
    a. **Download and Load the Dataset:**
        i. Follow the dataset link to download the data.
        ii. Use appropriate PyTorch Dataset classes or create custom ones if necessary.
    b. **Preprocessing:**
        i. Normalize the images using mean and standard deviation suitable for the dataset.
        ii. Resize images if needed to match input dimensions expected by your model.
    c. **Data Augmentation:**
        i. Apply transformations such as random horizontal flips, rotations, or color jitter to increase dataset variability.
3. **Model Building:**

   a. **Architecture:**
      i. Construct a CNN model incorporating the key components discussed in the notebook:
         1. Convolutional layers
         2. Pooling layers
         3. Activation functions (e.g., ReLU)
         4. Fully connected layers
         5. Dropout layers to prevent overfitting
   b. **Model Complexity:**
      i. Ensure the model is appropriately complex for the chosen dataset.

4. **Training:**
   a. **Hyperparameters:**
      i. Set initial hyperparameters like learning rate, batch size, number of epochs, etc.
   b. **Optimizer and Loss Function:**
      i. Use an appropriate optimizer (e.g., Adam, SGD) and loss function (e.g., CrossEntropyLoss).
   c. **Training Loop:**
      i. Implement the training loop with forward pass, loss computation, backward pass, and weight updates.
      ii. Include validation at the end of each epoch to monitor performance.
   d. **Learning Rate Scheduling:**
      i. Implement a learning rate scheduler to adjust the learning rate during training.
5. **Evaluation:**
   a. **Test the Model:**
      i. Evaluate your model on the test set.
      ii. Calculate accuracy and other relevant metrics (e.g., precision, recall, F1-score).
   b. **Confusion Matrix:**
      i. Plot a confusion matrix to visualize model performance across classes.
6. **Visualization:**
   a. **Filters and Feature Maps:**
      i. Visualize the learned filters of the convolutional layers.
      ii. Display feature maps after convolutional layers for sample images.
   b. **Training Curves:**
      i. Plot training and validation loss curves over epochs.

7. **Hyperparameter Tuning:**
    a. **Experimentation:**
        i. Try different hyperparameters to improve model performance.
        ii. Optionally, use GridSearchCV or similar methods for systematic tuning.
    b. **Document Findings:**
        i. Record how changes in hyperparameters affect the model's performance.
8. **Report:**
    a. **Summary:**
        i. Write a brief report summarizing your approach, experiments, and findings.
    b. **Analysis:**
        i. Discuss the model's performance and any challenges encountered.
        ii. Interpret the visualizations of filters and feature maps.

Datasets:

1. Fashion-MNIST

- **Description:** A dataset consisting of 70,000 grayscale images in 10 categories, each image is 28x28 pixels. The classes include different types of clothing items.
- **Link:** Fashion-MNIST GitHub

2. Street View House Numbers (SVHN)

- **Description:** A real-world image dataset for developing machine learning and object recognition algorithms with minimal requirement on data preprocessing and formatting. It has over 600,000 digit images (from 0 to 9).
- **Link:** SVHN Dataset

3. Intel Image Classification

- **Description:** Contains around 25,000 images categorized into six classes: buildings, forest, glacier, mountain, sea, and street.
- **Link:** Intel Image Classification Dataset: https://www.kaggle.com/datasets/puneet6060/intel-image-classification

4. Tiny ImageNet

- **Description:** A subset of the ImageNet dataset with 200 classes, each containing 500 training images, 50 validation images, and 50 test images.
- **Link:** Tiny ImageNet Visual Recognition Challenge https://www.kaggle.com/c/tiny-imagenet/data

Deliverables:

1. **Jupyter Notebook:**
    a. Well-documented code with explanations and comments.
    b. Outputs showing results, visualizations, and training progress.
2. **Report Document:**
    a. A concise report summarizing your methodology, experiments, results, and interpretations.
    b. Include graphs, charts, and images where relevant.