EXPERIMENT 5

OBJECTIVE : WAP to train and evaluate a convolutional neural network using Keras Library to classify MNIST fashion dataset. Demonstrate the effect of filter size, regularization, batch size and optimization algorithm on model performance.

DESCRIPTION OF MODEL

The model is a CNN(Convolutional Neural Network) implemented to classify the MNIST Fashion dataset. The Fashion-MNIST dataset, consists of 70,000 grayscale images (60,000 for training, 10,000 for testing) of 10 different fashion product classes, each image being 28x28 pixels.

Input size : (28,28,1) (1 channel for grayscale images)

- **Attempt 1 :**

2 Convolutional layers :

First Convolutional Layer: 32 , (3×3), activation='relu' .

Second Convolutional Layer: 64 ,(3×3), activation='relu'.

MaxPooling is applied after each convolutional layer .

Flatten layer : converts 2D vector to 1D for input to fully connected layer.

First fully connected layer : 128 neurons , activation='relu'.

Output fully connected layer : 10 neurons , activation='softmax'.

Batch size : 32 , 64 ,128 , Optimisation Algorithm : Adam , Stochastic gradient descent with momentum , Regularisation : L1 , L2 .

- **Attempt 2 :**

Convolutional Blocks : 2 (5 convolutional layers)

• Block 1 : 2 Conv2D(32,(3,3)) layers ,activation = ReLU and Batch Normalization.

• Block 2 : 2 Conv2D(64,(3,3)) layers ,activation = ReLU and Batch Normalization.

• Block 3 : 1 Conv2D(128,(3,3)) layers ,activation = ReLU and Batch Normalization. MaxPooling is applied after each convolutional block to reduce spatial dimention. Dropout Layers (0.2) after each convolutional block to prevent overfitting.

Flatten layer : converts 3D(h,w,channels) vector to 1D for input to fully connected layer. Fully connected layer :

• First layer : 512 neurons , activation= ReLU , batch normalisation. Dropout 0.2.

• Output fully connected layer : 10 neurons , activation='softmax'

Batch size cases : 64 ,128

Optimisation Algorithm cases : Adam , Stochastic gradient descent with momentum.
Regularisation : None , Used Batch Normalisation.

Loss function : Sparse categorical cross entropy

➢ Attempt 3 :

Convolutional Blocks : 4 (8 convolutional layers)

- Block 1 : 2 Conv2D(32, (3,3)) layers, activation = ReLU, Batch Normalization, MaxPooling(2,2), Dropout(0.3) .
- Block 2 : 2 Conv2D(64, (3,3)) layers, activation = ReLU, Batch Normalization, MaxPooling(2,2), Dropout(0.3) .
- Block 3 : 2 Conv2D(128, (3,3)) layers, activation = ReLU, Batch Normalization, MaxPooling(2,2), Dropout(0.4) .
- Block 4 : 2 Conv2D(256, (3,3)) layers, activation = ReLU, Batch Normalization.
- GlobalAveragePooling2D()(used because deeper network) : converts 3D(h,w,channels) vector to 1D for input to fully connected layer.

Fully connected layer :

- First layer : 512 neurons , activation= ReLU , batch normalisation.
- Dropout 0.4.
- Output fully connected layer : 10 neurons , activation='softmax'

Batch size : 128

Optimisation Algorithm : Adam

Regularisation : None , Used Batch Normalisation.

Loss function : Sparse categorical cross entropy

PYTHON IMPLEMENTATION

"""8 conv layers 35 epochs : train accuracy -95.40 , : test accuracy - 93.77 """

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

```
import matplotlib.pyplot as plt

# Load Fashion MNIST dataset
(x_train, y_train), (x_test, y_test) = keras.datasets.fashion_mnist.load_data()

# Normalize images to [0,1] range
x_train, x_test = x_train / 255.0, x_test / 255.0

# Reshape for CNN (Adding channel dimension)
x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)

# CNN Model with deeper architecture
model = keras.Sequential([
    layers.Conv2D(32, (3,3), padding='same', activation=None, input_shape=(28,28,1)),
    layers.BatchNormalization(),
    layers.Activation('relu'),

    layers.Conv2D(32, (3,3), padding='same', activation=None),
    layers.BatchNormalization(),
    layers.Activation('relu'),

    layers.MaxPooling2D((2,2)),
    layers.Dropout(0.3),

    layers.Conv2D(64, (3,3), padding='same', activation=None),
    layers.BatchNormalization(),
    layers.Activation('relu'),
```

```
layers.Conv2D(64, (3,3), padding='same', activation=None),
layers.BatchNormalization(),
layers.Activation('relu'),

layers.MaxPooling2D((2,2)),
layers.Dropout(0.3),

layers.Conv2D(128, (3,3), padding='same', activation=None),
layers.BatchNormalization(),
layers.Activation('relu'),

layers.Conv2D(128, (3,3), padding='same', activation=None),
layers.BatchNormalization(),
layers.Activation('relu'),

layers.MaxPooling2D((2,2)),
layers.Dropout(0.4),

layers.Conv2D(256, (3,3), padding='same', activation=None),
layers.BatchNormalization(),
layers.Activation('relu'),

layers.Conv2D(256, (3,3), padding='same', activation=None),
layers.BatchNormalization(),
layers.Activation('relu'),

layers.GlobalAveragePooling2D(),
```

```
    layers.Dense(512, activation=None),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.Dropout(0.4),

    layers.Dense(10, activation='softmax')
])

# Compile the model with reduced learning rate
model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.0003),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train the model with increased batch size
history=model.fit(x_train,y_train,epochs=35,batch_size=128, validation_data=(x_test, y_test))

# Evaluate test accuracy
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {test_acc:.4f}")

# Plot Accuracy & Loss Curves
plt.figure(figsize=(12,5))

plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
```

```
plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()


plt.subplot(1,2,2)

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val_loss'], label='Val Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.show()
```

DESCRIPTION OF CODE (8 convolutional layers)

The code implements Convolutional Neural Network to classify the MNIST Fashion dataset.

- keras.datasets.fashion_mnist.load_data() – downloads the MNIST fashion dataset with image and label pair and splits it into training and testing samples.
- Normalisation : Image pixel values are normalized in range [0,1].
- Reshape : (28,28,1) (height, width, channels) format is input format for CNN

- CNN Model :
  - Block 1 : 2 Conv2D(32, (3,3)) layers, activation = ReLU, Batch Normalization, MaxPooling(2,2), Dropout(0.3) .
  - Block 2 : 2 Conv2D(64, (3,3)) layers, activation = ReLU, Batch Normalization, MaxPooling(2,2), Dropout(0.3) .
  - Block 3 : 2 Conv2D(128, (3,3)) layers, activation = ReLU, Batch Normalization, MaxPooling(2,2), Dropout(0.4) .
  - Block 4 : 2 Conv2D(256, (3,3)) layers, activation = ReLU, Batch Normalization.
  - GlobalAveragePooling2D()(used because deeper network) : converts 3D(h,w,channels) vector to 1D for input to fully connected layer.
  - Fully connected layer :
  - First layer : 512 neurons , activation= ReLU , batch normalisation.
  - Dropout 0.4.
  - Output fully connected layer : 10 neurons , activation='softmax'

- Loss is calculated using the sparse cross entropy loss function , optimizer is used to updates the weights of the model to minimize the loss function. It adjusts the weights using gradient descent and backpropagation to improve the model's accuracy.
- model.compile() - function prepares the model for training , specifies optimiser ,loss function and metrics.
- model.fit() – Trains the model for 35 epochs and 128 batch size , and at end of each epoch it checks performance on test set.
- model.evaluate() – Evaluates the accuracy of the model.
- Accuracy curve , loss curve and confusion matrix are plotted to evaluate the performance of the model .

PERFORMANCE EVALUATION

- Performance of the model is evaluated using Accuracy curve , loss curve and confusion matrix .
- Maximum accuracy achieved **is 93.77 %**.
- Attempt 1 :
    - The maximum accuracy achieved for 2 Convolutional layers was less , maximum was **91.06 %** .
    - For Batch size = 64 , filter size = (3,3),(3,3) , Optimiser algorithm = Adam , Regularisation = L2(0.0001) .
    - Cases : batch size[32 ,64 ,128] , optimisers [Adam , SGD with momentum] , regularisers [L1 , L2].
- Attempt 2 :
    - The maximum accuracy achieved for 5 convolutional layers **93.44 %** for Adam optimizer and batch size 64 .
    - Cases : batch size[64 , 128] , optimisers [Adam , SGD with momentum]

- **Attempt 3** : (*Best case , 8 convolutional layers*)
    - The maximum accuracy achieved for **8** convolutional layers **93.77 %** for Adam optimizer and batch size 128 for 35 epochs .
    - At 90 epochs accuracy achieved is **94.46 %** but model slightly overfits .
    - Single case : Batch size – 128 , optimizer – Adam .

MY COMMENTS

- The accuracy increases by increasing the number of convolutional layers from 2 to 5 and slightly increases for 5 to 8 .
- Maximum accuracy achieved is **93.77 %** for 8 convolutional layers at 35 epochs , as we train model for more number of epochs , the model starts overfitting.
- For 8 convolutional layers , at 90 epochs accuracy is **94.46 %** but model **slightly overfits**.
- Further accuracy can be increased by using more number of convolutional layers , increasing number of filters in convolutional layers , varying different kernel sizes , manipulating dropout , using a different pooling methods .