CM6

Classification with Convolutional Neural Networks

- CNN-based Approach

a) Model 1 (Basic Model)– 2 Convolutional Layers and 1 Fully connected Layer

1) **Conv Layer 1** -> 32 filters of size 3*3 and relu activation function with 28*28*1 input
2) **Max Pool**: 2*2(Pooling layers are added after convolution layers to reduce the size of feature maps outputted by convolutional layer)
3) **Conv Layer 2** -> 64 filters of size 3*3 and relu activation
4) **FC layer** -> 5 output classes with softmax activation

Pooling Layers- Pooling layers are added after convolution layers to reduce the size of feature maps outputted by convolutional layer. It helps in approximating the outputs of a layer by aggregating nearby values.

Loss Function – Categorical Cross Entropy is used as it's widely used for multiclass classification problems and as our output labels are categorical and have been one-hot encoded.

Optimizer – Adam optimizer is used as it is an extension to the classic stochastic gradient descent and is very robust to large datasets. It converges quickly achieves good results.

Activation Functions -

Relu activation is computationally simpler and leads to faster convergence of the model unlike sigmoid activation which tends to saturate and hence is used in the intermediate layers.It also helps to take care of vanishing gradient problem.

Softmax activation function is used at the output as it's a popular choice for multiclass classification and goes well with categorical cross entropy loss function.

```
model1 = Sequential()
model1.add(Conv2D(32, (2, 2), activation='relu', input_shape=(28, 28, 1)))
model1.add(MaxPooling2D((2, 2)))
model1.add(Conv2D(64, (2, 2), activation='relu'))
model1.add(Flatten())
model1.add(Dense(5, activation='softmax'))
# compile model
model1.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
model1_fit = model1.fit(x_train, y_train, validation_split=0.2, batch_size=128,epochs=30)
model1.summary()
```

```
loss, accuracy = model1.evaluate(x_test,y_test)
print(loss, accuracy)
```

The model was trained for 30 epochs with batch size of 128 with validation split=0.2

Model 1	Training Accuracy	Test Accuracy
	91.69%	83.49%

b) Model 2 (Improved Model)– 3 Convolutional Layers and 2 Fully connected Layers

1) **Conv Layer 1** -> 64 filters of size 3*3 and relu activation function with 28*28*1 input shape with padding='same'
2) **Max Pool** -> 2*2
3) **Dropout** -> 0.25 (25 % of the nodes are dropped out randomly from the neural network)
4) **Conv Layer 2** -> 64 filters of size 3*3 and relu activation
5) **Max Pool** -> 2*2
6) **Dropout** -> 0.25 (25 % of the nodes are dropped out randomly from the neural network)
7) **Conv Layer 3** -> 64 filters of size 3*3 and relu activation
8) **Dropout** -> 0.25
9) **FC Layer** -> 64 hidden neurons with relu activation
10) **Dropout** -> 0.25
11) **FC Layer** -> 5 output classes with softmax activation

Categorical Cross Entropy loss function, Adam optimizer, Relu activation function in the intermediate convolutional and dense layers and Softmax activation is used at the output FC layer for the same reasons as Model 1.

The main difference in the architecture when compared to Model 1 are:

1) Additional convolution layer and fully connected layer with 64 hidden neurons are added and the network is made deeper for better learning and generalization with depth.
2) Each of the convolutional layers and FC layers are followed by Max Pooling and Dropout Layers.

   Pooling layers are added after convolution layers to reduce the size of feature maps outputted by convolutional layer. It helps in approximating the outputs of a layer by aggregating nearby values.

   In the **dropout layers**, a few neurons are dropped from the neural network during training process resulting in reduced size of the model.
   As a result the model has parameters to learn and prevents the **overfitting** that happens in the fully connected networks.

3) Adding padding to the convolutional operation can often result in better model performance, as more of the input image of feature maps are given an opportunity to participate or contribute to the output. By default, the convolutional operation uses 'valid'padding, which means that convolutions are only applied where possible. This can be changed to 'same' padding so that zero values are added around the input (zero padding) such that the output has the same size as the input.

```
model2 = Sequential()
model2.add(Conv2D(64, (3, 3),padding='same', activation='relu', input_shape=(28, 28, 1)))
model2.add(MaxPooling2D((2, 2)))
model2.add(Dropout(0.25))
model2.add(Conv2D(64, (3, 3), activation='relu'))
model2.add(MaxPooling2D((2, 2)))
model2.add(Dropout(0.25))
model2.add(Conv2D(64, (3, 3), activation='relu'))
model2.add(Dropout(0.25))
model2.add(Flatten())
model2.add(Dense(64, activation='relu'))
model2.add(Dropout(0.25))
model2.add(Dense(5, activation='softmax'))
# compile model
model2.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
model2_fit = model2.fit(x_train, y_train, validation_split=0.2, batch_size=128,epochs=30)
model2.summary()
```

The model was trained for 30 epochs with batch size of 128

Model 2	Training Accuracy	Test Accuracy
	88.89%	89.68%

- Other Deep Learning model variant approach

c) Model 3-Fully connected Feed Forward Deep NN Model-5 FC layers

1) FC Layer 1- 128 hidden neurons with relu activation
2) FC layer 2 – 128 hidden neurons with relu activation
3) FC layer 3 – 128 hidden neurons with relu activation
4) FC layer 4 – 128 hidden neurons with relu activation
5) FC layer 5 – 5 output classes with softmax activation

Unlike CNN, this model doesn't have convolutional layers, max pool layers or dropout layers. It utilizes relu activation functions in the intermediate layers and softmax activation at the output layers for the same reasons as model 1 and model 2.

```
model3=Sequential()
model3.add(Flatten())
model3.add(Dense(128, activation='relu'))
model3.add(Dense(128, activation='relu'))
model3.add(Dense(128, activation='relu'))
model3.add(Dense(128, activation='relu'))
model3.add(Dense(5, activation='softmax'))
model3.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
model3_fit = model3.fit(x_train, y_train, epochs=30, batch_size=128, validation_split=0.2)
model3.summary()
```

```
loss, accuracy = model3.evaluate(x_test,y_test)
print(loss, accuracy)
```

The model was trained for 30 epochs with batch size of 128

Model 3	Training Accuracy	Test Accuracy
	93.46%	88.97%