CSC 722 - Machine Learning Fundamentals

CNN Team Project

Team Members:

1. Dheeraj Avadhutha - 101129707
2. Pranav Reddy Dareddy - 101134583
3. Vamsi Yadala - 101130145
4. Sri Mukund Kadiyala - 101134587
5. Chenchaiah Mekalathuru - 101165473
6. Sai Krishna Yadav Madiboyena - 101132195
7. Shanmukh Sai Madhu - 101162913
8. Pavan Kumar Balli - 101136776
9. Sai Rajesh Chittavarjula - 101179899
10. Rohit Mareddy - 101134589


```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Create a Convolutional Neural Network (CNN) model
cnn_model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
])
```

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical


# Load and preprocess the MNIST dataset
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
train_images = train_images.reshape(-1, 28, 28, 1) / 255.0
test_images = test_images.reshape(-1, 28, 28, 1) / 255.0
train_labels = to_categorical(train_labels, num_classes=10)
test_labels = to_categorical(test_labels, num_classes=10)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

```
# Display model summary
cnn_model.summary()

from tensorflow.keras.layers import MaxPooling2D
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2  (None, 13, 13, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 5, 5, 64)          0
 g2D)

 conv2d_2 (Conv2D)           (None, 3, 3, 64)          36928

 flatten (Flatten)           (None, 576)               0

 dense (Dense)               (None, 64)                36928

 dense_1 (Dense)             (None, 10)                650
```

```
=================================================================
Total params: 93322 (364.54 KB)
Trainable params: 93322 (364.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
# Example of max pooling layer
max_pool_layer = MaxPooling2D(pool_size=(2, 2))

from tensorflow.keras.layers import Dense


# Fully connected layer followed by softmax
fc_layer = Dense(64, activation='relu')
softmax_layer = Dense(10, activation='softmax')


# Compile and train the CNN model
cnn_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = cnn_model.fit(train_images, train_labels, epochs=10, batch_size=128, validation_data=(test_images, test_labels))
```

```
Epoch 1/10
469/469 [==============================] - 57s 118ms/step - loss: 0.2497 - accuracy: 0.9241 - val_loss: 0.0660 - val_accurac
Epoch 2/10
469/469 [==============================] - 48s 102ms/step - loss: 0.0627 - accuracy: 0.9804 - val_loss: 0.0514 - val_accurac
Epoch 3/10
469/469 [==============================] - 51s 108ms/step - loss: 0.0426 - accuracy: 0.9865 - val_loss: 0.0347 - val_accurac
Epoch 4/10
469/469 [==============================] - 49s 104ms/step - loss: 0.0334 - accuracy: 0.9898 - val_loss: 0.0303 - val_accurac
Epoch 5/10
469/469 [==============================] - 49s 104ms/step - loss: 0.0278 - accuracy: 0.9910 - val_loss: 0.0373 - val_accurac
Epoch 6/10
469/469 [==============================] - 49s 104ms/step - loss: 0.0223 - accuracy: 0.9931 - val_loss: 0.0336 - val_accurac
Epoch 7/10
469/469 [==============================] - 50s 107ms/step - loss: 0.0193 - accuracy: 0.9937 - val_loss: 0.0293 - val_accurac
Epoch 8/10
469/469 [==============================] - 53s 113ms/step - loss: 0.0150 - accuracy: 0.9952 - val_loss: 0.0289 - val_accurac
Epoch 9/10
469/469 [==============================] - 50s 107ms/step - loss: 0.0136 - accuracy: 0.9956 - val_loss: 0.0319 - val_accurac
Epoch 10/10
469/469 [==============================] - 47s 101ms/step - loss: 0.0098 - accuracy: 0.9969 - val_loss: 0.0284 - val_accurac
```

```
# Evaluate the model
test_loss, test_accuracy = cnn_model.evaluate(test_images, test_labels)
print(f'Test Accuracy: {test_accuracy}')

from sklearn.model_selection import KFold
from sklearn.metrics import confusion_matrix
```

```
313/313 [==============================] - 3s 9ms/step - loss: 0.0284 - accuracy: 0.9919
Test Accuracy: 0.9919000267982483
```

```
# Use KFold for cross-validation
kf = KFold(n_splits=5, shuffle=True)
for train_index, test_index in kf.split(train_images):
    X_train_fold, X_val_fold = train_images[train_index], train_images[test_index]
    y_train_fold, y_val_fold = train_labels[train_index], train_labels[test_index]

    # Train and evaluate model for each fold
    cnn_model.fit(X_train_fold, y_train_fold)
    val_loss, val_accuracy = cnn_model.evaluate(X_val_fold, y_val_fold)
    print(f'Validation Accuracy: {val_accuracy}')

    # Generate confusion matrix
    predictions = cnn_model.predict(X_val_fold)
    cm = confusion_matrix(y_val_fold.argmax(axis=1), predictions.argmax(axis=1))
    print(cm)
```

```
1500/1500 [==============================] - 45s 30ms/step - loss: 0.0325 - accuracy: 0.9890
375/375 [==============================] - 3s 9ms/step - loss: 0.0274 - accuracy: 0.9908
Validation Accuracy: 0.9908333420753479
375/375 [==============================] - 4s 10ms/step
[[1155    0    0    0    1    0    1    0    2    1]
 [   0 1401    0    0    0    0    0    7    4    0]
 [   2    1 1151    1    0    0    0    5    1    0]
 [   2    0    2 1229    0    6    0    2    2    2]
 [   0    1    0    0 1094    0    1    4    6   13]
```

```
 [   0    0    1    1    0 1101    1    0    1    2]
 [   3    2    1    0    1    0 1159    0    2    0]
 [   0    1    3    0    0    0    0 1220    2    0]
 [   2    0    0    0    0    4    1    1 1169    2]
 [   0    0    0    1    2    2    0    3    4 1211]]
1500/1500 [==============================] - 44s 29ms/step - loss: 0.0209 - accuracy: 0.9934
375/375 [==============================] - 5s 12ms/step - loss: 0.0180 - accuracy: 0.9939
Validation Accuracy: 0.9939166903495789
375/375 [==============================] - 3s 9ms/step
[[1191    0    0    0    0    0    0    0    0    1]
 [   0 1340    1    0    1    0    1    5    0    1]
 [   0    1 1155    2    0    0    0    3    0    1]
 [   1    0    4 1238    0    4    0    1    2    1]
 [   0    1    0    0 1155    0    0    2    0   10]
 [   0    0    0    1    0 1058    3    1    1    1]
 [   3    0    0    0    0    1 1191    0    0    0]
 [   0    0    1    0    0    0    0 1273    0    1]
 [   1    1    0    0    3    2    0    0 1175    8]
 [   0    0    0    0    2    0    0    0    0 1151]]
1500/1500 [==============================] - 44s 30ms/step - loss: 0.0177 - accuracy: 0.9941
375/375 [==============================] - 3s 9ms/step - loss: 0.0206 - accuracy: 0.9937
Validation Accuracy: 0.9937499761581421
375/375 [==============================] - 3s 9ms/step
[[1187    0    2    1    0    1    0    0    0    0]
 [   0 1256    0    0    0    0    0    3    0    0]
 [   0    2 1189   10    0    0    0    3    0    0]
 [   0    1    0 1222    0    1    0    1    0    0]
 [   0    1    0    0 1194    0    2    4    0    1]
 [   0    0    0    3    0 1121    0    0    0    0]
 [   0    1    0    1    1    5 1183    0    8    0]
 [   0    0    0    0    0    0    0 1258    0    0]
 [   0    0    2    2    0    0    0    0 1128    1]
 [   0    2    0    1    3    1    0   11    0 1187]]
1500/1500 [==============================] - 42s 28ms/step - loss: 0.0157 - accuracy: 0.9951
375/375 [==============================] - 4s 10ms/step - loss: 0.0079 - accuracy: 0.9974
Validation Accuracy: 0.9974166750907898
375/375 [==============================] - 3s 9ms/step
[[1174    0    0    0    0    0    2    0    1    0]
 [   0 1394    0    0    1    0    0    0    1    0]
 [   0    0 1213    0    1    0    0    0    0    0]
 [   0    0    1 1189    0    0    0    0    0    0]
 [   0    0    0    0 1217    0    1    0    0    1]
 [   1    0    0    1    0 1062    1    0    1    0]
 [   1    0    0    0    1    1 1137    0    0    0]
 [   0    5    1    1    1    0    0 1274    0    1]
 [   0    0    0    0    0    0    1    0 1115    1]
 [   0    2    0    0    1    0    0    1    0 1194]]
1500/1500 [==============================] - 44s 29ms/step - loss: 0.0111 - accuracy: 0.9963
375/375 [==============================] - 3s 9ms/step - loss: 0.0064 - accuracy: 0.9976
```