Report on Image Classification using Deep Learning on CIFAR-10 Dataset

Thursday, 06.06.2024

—

Introduction

This report details the implementation and performance evaluation of an image classification model using the CIFAR-10 dataset. The model leverages Deep Learning techniques and was trained and evaluated using various optimization algorithms and loss functions. The aim was to compare their effectiveness based on several performance metrics.

Model Architecture

The model architecture is a Convolutional Neural Network (CNN) with the following layers:

- **Conv2D**: 32 filters, kernel size (3, 3), ReLU activation
- **MaxPooling2D**: pool size (2, 2)
- **Conv2D**: 64 filters, kernel size (3, 3), ReLU activation
- **MaxPooling2D**: pool size (2, 2)
- **Conv2D**: 128 filters, kernel size (3, 3), ReLU activation
- **Flatten**
- **Dense**: 512 units, ReLU activation
- **Dropout**: 0.5
- **Dense**: 10 units, softmax activation

Experimental Setup

Experiments were conducted using three different optimization algorithms (Adam, SGD, RMSProp) and two loss functions (Categorical Crossentropy, Mean Squared Error). Each combination was trained for 10 epochs, and the performance was evaluated using the following metrics:

- **Accuracy**
- **Precision**
- **Recall**

- **F1 Score**
- **Specificity**

Results

The following tables summarize the performance of each model configuration:

Adam Optimizer

1. Loss Function: Categorical Crossentropy
    - **Accuracy**: 73.09%
    - **Precision**: 0.7369
    - **Recall**: 0.7309
    - **F1 Score**: 0.7310
    - **Specificity**: Varies from 0.549 to 0.901

Confusion Matrix:

```
[[824  32  28  16  23   6   3   8  42  18]
 [ 17 901   6   9   4   4   7   0  17  35]
 [ 72  15 658  48  78  40  55  17   9   8]
 [ 22  23  97 549  79 131  52  22  14  11]
 [ 28   4  75  49 733  31  26  43   9   2]
 [ 16   8  72 162  59 620  20  33   4   6]
 [  9   7  54  63  45  16 790   8   6   2]
 [ 25   5  47  39  78  52   2 738   3  11]
 [109  41  16  12  12   4   2   2 788  14]
 [ 52 147  15  21  10  11   5   9  22 708]]
```

2. Loss Function: Mean Squared Error
    - **Accuracy**: 73.36%
    - **Precision**: 0.7327
    - **Recall**: 0.7336
    - **F1 Score**: 0.7316
    - **Specificity**: Varies from 0.542 to 0.862

Confusion Matrix:

```
[[757  31  36  16  17   3   9  15  92  24]
 [ 10 860   5   4   5   5  20   6  24  61]
 [ 56   9 606  53  68  77  64  33  24  10]
 [ 15  13  64 542  43 185  57  46  15  20]
 [ 16   4  58  64 627  52  77  84  15   3]
 [ 11   6  56 143  31 645  22  69   7  10]
 [  3   6  33  43  17  35 840  11   5   7]
 [ 13   3  19  32  34  66  11 809   4   9]
 [ 34  36  10  10   4  10   5   7 862  22]
 [ 29  89   7  10   5  10  12  15  35 788]]
```

SGD Optimizer

1. Loss Function: Categorical Crossentropy
    - **Accuracy**: 48.51%
    - **Precision**: 0.5603
    - **Recall**: 0.4851
    - **F1 Score**: 0.4881
    - **Specificity**: Varies from 0.288 to 0.895

Confusion Matrix:

```
[[430  14  94   4  23   6   3   3 411  12]
 [ 59 508  24   6   9   9   6   6 284  89]
 [ 72   6 576  42 118  43  21  18  90  14]
 [ 27   8 266 288 123 124  31  20  95  18]
 [ 34   2 265  27 523  22  26  35  63   3]
 [ 13   4 275 154  88 340  16  40  64   6]
 [  9  14 165  47 231  19 442  11  49  13]
```

```
[ 29   8 167  45 139  70   4 463  49  26]
[ 39  12  25   5   4   6   4   2 895   8]
[ 41 101  46  16  17   9   8  19 357 386]]
```

2. Loss Function: Mean Squared Error
    - **Accuracy**: 16.28%
    - **Precision**: 0.1155
    - **Recall**: 0.1628
    - **F1 Score**: 0.0987
    - **Specificity**: Varies from 0.0 to 0.901

Confusion Matrix:

```
[[657   0   0   1   0  48   3   0 259  32]
 [485   0   0   0   0 126  13   0 277  99]
 [558   0   0   3   0 244   9   0 139  47]
 [476   0   1   7   1 302  11   1 135  66]
 [489   0   1   8   0 318  31   2  93  58]
 [417   0   0   4   1 317  16   0 187  58]
 [374   0   0   5   0 476  27   0  54  64]
 [578   0   0   6   0 162   9   0 124 121]
 [445   0   0   0   0  62   1   0 437  55]
 [511   0   0   0   0  54   9   1 242 183]]
```

    -

RMSProp Optimizer

1. Loss Function: Categorical Crossentropy
    - **Accuracy**: 72.49%
    - **Precision**: 0.7324
    - **Recall**: 0.7249
    - **F1 Score**: 0.7233
    - **Specificity**: Varies from 0.458 to 0.841

Confusion Matrix:

```
[[746  21  21  24  27   6  12  13  81  49]
 [ 17 841   2  10   5   6   7   3  20  89]
 [ 95   7 458 113 106  81  72  35  11  22]
 [ 25   5  18 617  66 130  61  42   9  27]
 [ 18   3  30  83 736  30  39  45   6  10]
 [ 17   4  17 195  48 628  19  50   4  18]
 [  3   4  15  69  55  32 799  10   4   9]
 [ 10   1  11  60  58  58  10 764   5  23]
 [ 54  34   5  16   8   6   7   6 822  42]
 [ 15  71   5  17   7   4   6  10  27 838]]
```

2. Loss Function: Mean Squared Error
    - **Accuracy**: 71.48%
    - **Precision**: 0.7329
    - **Recall**: 0.7148
    - **F1 Score**: 0.7144
    - **Specificity**: Varies from 0.509 to 0.901

Confusion Matrix:

```
[[778   8  85   4  34   2  23   1  44  21]
 [ 27 778  14   3  14   4  29   4  40  87]
 [ 44   2 657  32 107  27 103  10  11   7]
 [ 13   2  99 509 106  84 125  16  22  24]
 [ 15   0  74  35 744  11  90  16  13   2]
 [ 13   3 125 148  93 511  72  17  10   8]
 [  4   0  26  25  29   4 901   2   9   0]
 [ 11   3  66  29 167  43  32 631   6  12]
 [ 62  10  33   8  10   3  12   1 843  18]
```

```
 [ 40  57  21   9  13   6  19   4  35 796]]
```

Conclusion

- The best-performing model used the **Adam optimizer with categorical crossentropy loss**, achieving an accuracy of 73.09%.
- **SGD with mean squared error** performed poorly, indicating a mismatch for the CIFAR-10 classification task.
- Future improvements could include deeper networks, learning rate adjustments, and data augmentation techniques to enhance model robustness.

References

- CIFAR-10 Dataset: Link
- TensorFlow and Keras Documentation

Appendix: Full Code for CIFAR-10 Image Classification

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, Dropout
from tensorflow.keras.optimizers import Adam, SGD, RMSprop
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,
recall_score, f1_score

# Load Cifar-10 dataset
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Normalize the data
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

# Convert labels to one-hot encoding
y_train = to_categorical(y_train, 10)
```

```
y_test = to_categorical(y_test, 10)

# Build the model function
def build_model(optimizer, loss):
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
        MaxPooling2D((2, 2)),
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Conv2D(128, (3, 3), activation='relu'),
        Flatten(),
        Dense(512, activation='relu'),
        Dropout(0.5),
        Dense(10, activation='softmax')
    ])

    model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])
    return model

# Initialize optimizers and loss functions
optimizers = ['adam', 'sgd', 'rmsprop']
loss_functions = ['categorical_crossentropy', 'mean_squared_error']

# Clear previous history from memory to avoid confusion
tf.keras.backend.clear_session()

# Function to evaluate the model and compute metrics
def evaluate_model(model):
    y_pred = np.argmax(model.predict(x_test), axis=1)
    y_true = np.argmax(y_test, axis=1)

    conf_matrix = confusion_matrix(y_true, y_pred)
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='macro')
    recall = recall_score(y_true, y_pred, average='macro')
    f1 = f1_score(y_true, y_pred, average='macro')
    specificity = np.diag(conf_matrix) / np.sum(conf_matrix, axis=1)

    return conf_matrix, accuracy, precision, recall, f1, specificity

history_list = []
metrics_list = []
```

```
for opt_name in optimizers:
    for loss_name in loss_functions:
        print(f"\nTraining with Optimizer: {opt_name}, Loss Function: {loss_name}\n")

        if opt_name == 'adam':
            optimizer = tf.keras.optimizers.legacy.Adam()
        elif opt_name == 'sgd':
            optimizer = tf.keras.optimizers.legacy.SGD()
        elif opt_name == 'rmsprop':
            optimizer = tf.keras.optimizers.legacy.RMSprop()

        model = build_model(optimizer, loss_name)

        history = model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test),
batch_size=64, verbose=2)
        history_list.append((opt_name, loss_name, history))

        # Evaluate model and collect metrics
        conf_matrix, accuracy, precision, recall, f1, specificity = evaluate_model(model)
        metrics_list.append((opt_name, loss_name, conf_matrix, accuracy, precision,
recall, f1, specificity))

        # Clear session to avoid conflicts in subsequent trainings
        tf.keras.backend.clear_session()

# Displaying metrics
for opt_name, loss_name, conf_matrix, accuracy, precision, recall, f1, specificity in
metrics_list:
    print(f"Optimizer: {opt_name}, Loss Function: {loss_name}")
    print(f"Confusion Matrix:\n{conf_matrix}")
    print(f"Accuracy: {accuracy}")
    print(f"Precision: {precision}")
    print(f"Recall: {recall}")
    print(f"F1 Score: {f1}")
    print(f"Specificity: {specificity}")
    print("\n")
```