DL ASSIGNMENT 2

Marrium Jilani - 20K-1748

October 22, 2023

1 Task 1:Skin Cancer Classification Using CNN - Experimental Results

1.1 Introduction

The task involved classifying skin cancer images into seven different classes using deep learning techniques. The dataset was divided into training and testing sets, with an 80-20 split. The objective was to achieve a classification accuracy of at least 75

1.2 Data Preprocessing

The dataset was obtained from kaggle and used directly for processing.The dataset was preprocessed to handle class imbalances. Random oversampling was applied to balance the classes.
"RandomOverSampler" is an oversampling technique that generates additional samples for the minority class by randomly selecting and duplicating existing samples from the minority class until it matches the number of samples in the majority class. This is a straightforward approach to balance class distribution.

Training Samples: To display samples from each class, the 'show_samples' function was used.

1.3 Model Architecture

The Convolutional Neural Network (CNN) architecture used for skin cancer classification is designed to capture relevant features from the input images and make accurate predictions. Here is an overview of the key components of the model:

Input Layer:

1. The input layer is configured to accept images with a shape of (28, 28, 3), which corresponds to a 28x28-pixel image with three color channels (RGB).
2. Activation Function: Rectified Linear Unit (ReLU) activation is applied to introduce non-linearity.

Convolutional Layers:

1. The model consists of multiple convolutional layers.
2. The first convolutional layer has 16 filters/kernels with a size of (3, 3).
3. Subsequent convolutional layers increase the number of filters from 32 to 128.
4. Activation Function: ReLU is applied after each convolution operation.

Pooling Layers:

1. Max-pooling layers with a pool size of (2, 2) follow the convolutional layers.
2. Max-pooling reduces the spatial dimensions of the feature maps while retaining essential information.

Batch Normalization:

1. Batch normalization layers are added after convolutional and dense layers.

2. Batch normalization helps stabilize and speed up the training process by normalizing the inputs.

Fully Connected Layers:

1. The Flatten layer is used to convert the 2D feature maps from the convolutional layers into a 1D vector.
2. The model includes fully connected (dense) layers with varying numbers of neurons (e.g., 256, 128, 64, and 32).
3. Activation Function: ReLU is applied to the neurons in these layers.
4. Dropout Layers: Dropout layers with a dropout rate of 0.2 are included to prevent overfitting.

Output Layer:

1. The output layer is composed of seven neurons, one for each class of skin cancer.
2. Activation Function: Softmax activation is used to transform the network's raw output into class probabilities.

1.4 Training and Results

The dataset was split into training and validation sets. The model was trained for 40 epochs with a batch size of 128. Early stopping was implemented to prevent overfitting. The training process resulted in an accuracy improvement from 80% to 95%.

1.5 Evaluation and Metrics

The trained model was evaluated on the test data, and the following metrics were calculated: accuracy, precision, recall, and F1-score. The model achieved the required accuracy of at least 75

1.6 Conclusion

The experiments successfully achieved the classification accuracy objective. The CNN architecture, along with data preprocessing and image preprocessing, played a significant role in achieving the desired results.

2 Task 2: Stock Price Prediction Using RNN - Experimental Results

2.1 Introduction

Task 2 focuses on the prediction of stock prices using Recurrent Neural Networks (RNN) and historical stock price data. This experiment aims to forecast future stock prices based on past Open, High, Low, Close (OHLC) prices, and trading volumes. The key steps in this task include data cleaning, normalization, sequence transformation, hyperparameter tuning, model training, and deployment in a containerized application.

2.2 Data Preprocessing

The dataset was obtained from Yahoo Finance, and it consists of historical stock price data for a publicly traded company (in this case, "AAPL"). Data preprocessing involved the following steps:

2.2.1 Handling Missing Values

Missing values in the dataset were addressed by filling them with the mean value of the respective columns. This approach helps in preserving data integrity.

2.2.2 Normalization

Two methods for data normalization were applied:

1. **Min-Max Scaling:** This method scales the data to a specific range, typically between 0 and 1. It is well-suited for features with well-defined boundaries, such as stock prices.
2. **Standardization (Z-Score Scaling):** Data was scaled to have a mean of 0 and a standard deviation of 1. This method is more appropriate for features that may not have clear boundaries.

2.3 Sequence-to-Sequence Transformation

In this step, the time series data was transformed into a sequence-to-sequence format. The goal was to create input sequences containing historical OHLC prices and volumes to predict future Close prices. The length of the input sequence can be adjusted as needed. For this task it was set to be 10.

2.4 RNN Architecture and Hyperparameter Tuning

2.4.1 RNN Architecture

The chosen RNN architecture for this task is based on Long Short-Term Memory (LSTM), a type of RNN known for handling sequential data effectively. LSTM layers are stacked to capture patterns and dependencies in the time series data.

2.4.2 Hyperparameter Tuning

The following hyperparameters were fine-tuned during the experimentation:

- Number of LSTM units
- Batch size
- Learning rate
- Number of epochs

The model architecture and hyperparameter tuning play a crucial role in achieving accurate stock price predictions.

2.5 Model Training and Results

The dataset was divided into training, validation, and test sets. The LSTM-based RNN model was trained using the training data while monitoring performance on the validation set. Early stopping was applied to prevent overfitting.

2.6 Evaluation and Metrics

The model's performance was evaluated on the test data using the Mean Squared Error (MSE) metric. MSE measures the average squared difference between the predicted and actual stock prices. A lower MSE indicates better predictive accuracy. In our case the MSE was approximately 0.01145 which is a very good value.

2.7 Conclusion

The experiment successfully demonstrates the prediction of stock prices using RNNs. The preprocessing steps, LSTM-based RNN architecture, and hyperparameter tuning collectively contribute to accurate stock price forecasts. This approach can be extended to various publicly traded companies and is valuable for financial analysis and decision-making.

3 Task 3: Facial Expression Recognition - Experimental Results

3.1 Introduction

In Task 3, the focus is on facial expression recognition using a dataset from Kaggle. The dataset contains images of different facial expressions, and the objective is to build a convolutional neural network (CNN) model for expression recognition. This section describes the technical details of the experiments performed.

3.2 Exploratory Data Analysis (EDA)

Exploratory data analysis was conducted to gain a better understanding of the dataset. The following aspects were explored:

3.2.1 Distribution of Facial Expression Labels

The distribution of facial expression labels was visualized to check for class imbalances in the data. It is crucial to understand if certain expressions are overrepresented or underrepresented.

3.2.2 Random Image Samples

To get a sense of the data, four random images from each expression class were displayed. This helps in observing the variations and nuances in the dataset.

3.3 CNN Model Architecture

A basic CNN model architecture was built for expression recognition. Several model architectures were experimented with, and an optimized architecture was selected. Hyperparameters, such as learning rate, were fine-tuned to achieve a decent baseline accuracy. The number of epoch was tweaked with until some substantial accuracy was reached for training set.

3.4 Model Evaluation

The performance of the CNN model was evaluated using appropriate metrics, including:

1. **Accuracy:** Accuracy is a standard metric that measures the overall correctness of predictions.
2. **Precision and Recall:** Precision measures the accuracy of positive predictions, while recall measures the ability to capture all positives.
3. **F1-Score:** The F1-score is the harmonic mean of precision and recall, providing a balanced measure of model performance.

To further test the model, it was given 10 random images and it predicted its class. That was displayed along with its original class.

3.5 Conclusion

The experiments in Task 3 focused on building a CNN model for facial expression recognition. EDA provided insights into the data distribution, and hyperparameter optimization enhanced model performance. The optimized model architecture achieved a decent baseline accuracy, and appropriate evaluation metrics were used to assess its performance.