**Generative Deep Learning**

**Siamese Convolutional Neural Networks for Authorship Verification**

**1. Introduction**

The goal of this project was to develop and implement a machine learning model to verify authorship of textual data. By employing Siamese Neural Networks (SNNs), the project aimed to analyze stylistic features of authors' writing to determine whether two text samples were written by the same author.

**2. Data Preparation and Project Flow**

**The project was divided into several phases to ensure a structured approach:**

**2.1 Dataset Preparation**

* **Dataset:** The "Fifty Victorian Era Novel Authorship Attribution" dataset was used.
* **Content:** The dataset included text samples from 50 authors with standardized lengths, designed for studying authorship attribution.
* **Dataset Splitting:** The dataset was split into 80% for training and 20% for testing using the train\_test\_split function with a fixed random state for consistent results.
* **Pair Creation:** From the training and testing data, pairs of texts were created using the create\_pairs function. A maximum of 10 authors were selected for training, with up to 20 text pairs generated per author.
* **Outputs:** The paired sequences were saved into files: Xtrain.npy, Ytrain.npy, Xtest.npy, and Ytest.npy. These files store the input sequences for training and testing (Xtrain and Xtest) and their binary labels (Ytrain and Ytest), indicating whether the paired texts were written by the same author (1) or different authors (0).

**2.2 Model Architecture Design**

* The Siamese Neural Network architecture was chosen for its ability to measure similarity between pairs of inputs.
* Key Components:
  1. **Character Embeddings:** Text inputs are tokenized, and each character is converted into a dense vector representation. This layer captures semantic and structural information about the characters.
  2. **First Convolutional Layer:** Extracts local features from small character sequences such as frequent character pairs, prefixes, or suffixes.
  3. **Second Convolutional Layer:** Builds on the extracted features from the first layer and detects more complex patterns that represent the author's overall writing style.
  4. **MaxPooling Layer:** Reduces dimensionality by selecting the most important features and discarding less relevant ones. It helps prevent overfitting.
  5. **LSTM Layer:** Captures long-term dependencies in the text. It learns sequential patterns and stylistic consistency throughout the writing.
  6. **Output Feature Vectors:** Converts the processed information into fixed-size vectors. These vectors summarize the author's stylistic features and are used for comparison.
  7. **Similarity Calculation:** Measures the distance between the output feature vectors of two texts using Euclidean distance. A smaller distance indicates higher similarity, suggesting the same author.

**2.3 Training Process**

* **Loss Function:** Contrastive Loss was used to minimize the distance between feature vectors of similar text pairs and maximize the distance for dissimilar pairs.
* **Optimizer:** Adam optimizer was employed for faster convergence.
* **Dataset Splits:** The data was divided into training, validation, and test sets.
* **Training Parameters:** Batch size, number of epochs, and learning rate were tuned for optimal performance.

**2.4 Testing and Validation**

* The model was tested on unseen text pairs to evaluate its accuracy in determining authorship similarity.
* Similarity Metric: Euclidean distance was used to measure the similarity between feature vectors.

**3. Results**

**Model Training:**

* **The model was trained on 10 authors, with 20 text samples per author.**
* **Training was conducted over 25 epochs, with a batch size of 512.**

**Outputs:**

1. **Author Processing Log:**

* A screenshot of a computer

  Description automatically generatedExample output from author processing:

1. **Model Summary:**

* A screenshot of a computer

  Description automatically generatedKey layers and parameters:

1. **Training Accuracy:**

* Example results for training and testing accuracy:

A screenshot of a computer program

Description automatically generated

* The model consistently achieved high testing accuracy above 90%, demonstrating its effectiveness in distinguishing between similar and dissimilar text pairs.

1. **Final Model Saved:**

* The model was saved **as lstm\_model.h5** for future use.

1. **New Testing Functionality**: A new function, predict\_author\_similarity, was implemented to directly test the model's predictions on individual text pairs. This function calculates the Euclidean distance between the feature vectors of two text samples and determines whether they were written by the same author.

def predict\_author\_similarity(model, tokenizer, text1, text2, max\_length=800):

    """

    Predict if two given texts are written by the same author.

    Args:

        model: Trained model.

        tokenizer: Tokenizer used during training.

        text1 (str): First text input.

        text2 (str): Second text input.

        max\_length (int): Maximum sequence length for padding.

    Returns:

        str: "Same author" or "Different authors".

    """

    # Tokenize and pad the texts

    seq1 = tokenizer.texts\_to\_sequences([text1])[0]

    seq2 = tokenizer.texts\_to\_sequences([text2])[0]

    seq1 = pad\_sequences([seq1], maxlen=max\_length)

    seq2 = pad\_sequences([seq2], maxlen=max\_length)

    # Predict using the two inputs separately

    prediction = model.predict([seq1, seq2], verbose=0)[0][0]  # Get the prediction (distance)

    print(f"Prediction : {prediction}")

    if prediction > 0.5:

        return "Same author"

    else:

        return "Different authors"

print("Text1 vs Text2:", predict\_author\_similarity(model, tokenizer, text1, text2))

print("Text1 vs Text3:", predict\_author\_similarity(model, tokenizer, text1, text3))

A black text on a white background

Description automatically generated

**Results from Testing:** Using the function, we tested the following pairs:

* Text1 vs. Text2: Prediction: 0.595 (Same author)
* Text1 vs. Text3: Prediction: 0.479 (Different authors)

These results indicate that the model can successfully differentiate between writing styles, even in complex scenarios. However, a threshold of 0.5 was used to decide authorship similarity, which may need fine-tuning for specific datasets or real-world applications.

**4. Discussion and Analysis:**

**Strengths of the Approach:**

1. The Siamese Neural Network effectively captured stylistic features, making it robust for authorship verification tasks.
2. The use of Euclidean distance as a similarity metric allowed for precise differentiation between authors.
3. The Victorian dataset provided a rich variety of styles, enhancing model generalization.
4. The addition of the predict\_author\_similarity function provided an intuitive and flexible way to validate the model's predictions on individual text pairs, ensuring practical usability.

**Challenges:**

1. Balancing the dataset to ensure equal representation of similar and dissimilar pairs.
2. Optimizing hyperparameters to prevent overfitting while maintaining high accuracy.
3. Fine-tuning the threshold value for authorship similarity predictions to better handle edge cases and improve performance on diverse datasets.

**Analysis of Results:**

1. The model achieved consistently high testing accuracy (above 90%), demonstrating its ability to generalize effectively.
2. A loss reduction over 25 epochs indicates effective learning of stylistic differences and similarities between authors.
3. The near-zero training accuracy in the last epochs suggests potential overfitting or unbalanced training data, highlighting the need for further investigation.
4. The implementation of the predict\_author\_similarity function revealed the model's real-world application potential by effectively testing unseen text pairs and accurately identifying whether they shared the same author.

**5. Conclusions:**

This project successfully demonstrated the effectiveness of Siamese Neural Networks for authorship verification. By analyzing stylistic features of text samples, the model achieved high accuracy and robustness, making it suitable for real-world applications such as cybersecurity, academic integrity checks, and forensic investigations. The addition of the predict\_author\_similarity function further validated the practical usability of the model, enabling on-the-fly evaluation of text pairs for authorship verification tasks. The results validate the potential of SNNs in verifying authorship and detecting similarities in textual data.

**6. References**

1. **Dataset: Fifty Victorian Era Novel Authorship Attribution Dataset (**[**Kaggle Link**](https://www.kaggle.com/datasets/luisfredgsousa/fifty-victorian-era-novel-authorship-attribution)**)**
2. **Paper: "A Robust Approach to Authorship Verification Using Siamese Deep Learning" (**[**Springer Link**](https://link.springer.com/content/pdf/10.1007/s10772-024-10110-y.pdf?)**)**
3. **Paper: "Siamese Convolutional Neural Networks for Authorship Verification" (**[**Stanford Report**](https://cs231n.stanford.edu/reports/2017/pdfs/801.pdf)**)**
4. **GitHub Repository: Authorship Attribution (**[**GitHub Link**](https://github.com/FernandoLpz/AuthorVerificiation)**)**