3 Modeling

3.1 NLP Processing and Vectorizing

The preprocessing of text data is critical to the performance of the models. The text of news articles is first cleaned to remove unnecessary characters and words that do not contribute to the understanding of the content. This includes stripping out special characters, numbers, and punctuations, followed by converting the text to lowercase to maintain consistency. Natural language toolkit (nltk) is utilized for tokenizing the text into individual words and removing stopwords—common words that are unlikely to contribute to text classification. Further, each word is lemmatized, reverting it to its base or dictionary form, which helps in reducing the complexity of the model without losing valuable information.

Words are then converted into numerical data through vectorization. A tokenizer is developed to create a dictionary of word indices, where each word is assigned a unique integer. Texts are transformed into sequences of these integers, and padded to ensure each input sequence is of the same length, a requirement for training neural networks.

3.2 Models

The ANN model designed for this project features a sequential architecture starting with an embedding layer, which transforms the vectorized words into dense vectors capturing semantic meanings. The network includes multiple dense layers with high neuron counts and ReLU activation to introduce non-linearity, enhancing the model's learning capability. Dropout layers are interspersed between dense layers to prevent overfitting by randomly setting a fraction of input units to 0 at each update during training time.

The CNN model, on the other hand, exploits the spatial structure of the data by applying convolutional layers that slide over input text vectors, capturing spatial hierarchies and relationships between words. This is followed by max-pooling layers that reduce dimensionality and computation, as well as help in making the detection of features invariant to scale and orientation. Like the ANN model, the CNN also incorporates dropout layers and dense layers towards the end to classify the input text into 'real' or 'fake'.

Both models are compiled with the Adam optimizer and categorical crossentropy loss function, which is suitable for binary classification tasks. They are trained with a validation set to monitor overfitting and underfitting, using metrics such as accuracy and loss. Training progress is visualized using accuracy and loss curves for both training and validation sets, providing insight into the model's performance and convergence behavior.

3.3 Text2Image

Beyond textual analysis, the project also incorporates a Text2Image feature directly integrated into the Svelte-based frontend, not as an experimental feature but as a functional component of the application. This feature uses OpenAI's API to generate images from text, enhancing the user interaction by providing visual representations of the classified articles. When a user submits a news article, the text is processed and classified by the backend model; concurrently, a relevant image is generated based on the article's content, thus merging visual AI capabilities with text classification.

The backend for this feature is managed by a Flask application which handles requests, processes the text for classification, and interacts with OpenAI's API to generate images. This integration not only demonstrates the practical application of machine learning and AI but also enhances the overall user experience by making the outcomes of the model more tangible and engaging.

The frontend to classify new articles looks like this: