Study

CNN

🔹 1. Convolutional Neural Network (CNN)

- **Layer used:** `Conv1D`
- **Purpose:** Extracts local features (e.g., n-gram-like patterns) from sequences of word embeddings, useful for capturing important phrases in text.

🔹 2. Bidirectional Long Short-Term Memory (BiLSTM)

- **Layer used:** `Bidirectional(LSTM)`
- **Purpose:** Processes the text both forward and backward to capture long-term dependencies in both directions, improving context understanding.

🔹 3. Embedding Layer

- **Layer used:** `Embedding`
- **Purpose:** Transforms tokenized words into dense vector representations. These vectors capture semantic relationships between words.

🔹 4. Regularization and Optimization

- **Dropout Layer:** Helps prevent overfitting by randomly turning off some neurons during training.
- **L2 Regularization:** Applied in the dense layer to further mitigate overfitting.
- **Adam Optimizer:** An adaptive learning rate optimization algorithm used to minimize loss during training.

🔹 5. Evaluation Metrics

- **Accuracy, Macro F1, Weighted F1, Confusion Matrix, and Classification Report:** Used to measure the performance of your multi-class classification model.

Concept	Details
Input	Word embeddings of size (200, 128)

Kernel (Filter) Shape	(5, 128) — looks at 5-word chunks
Output per filter	One number per window → Feature map
Filters	64 — each learns a different pattern
Output Shape after Conv1D	(196, 64)
Pooling	MaxPooling1D downsamples to (98, 64)
Activation	ReLU: introduces non-linearity
Algorithm used	Convolution (dot product), ReLU, backpropagation (with Adam optimizer)
Role in model	Extracts key phrase-level features before feeding to BiLSTM

BiLSTM

Component	Details
Model Layer	`Bidirectional(LSTM(64))`
Input Shape	Output from CNN: (98, 64)
Output Shape	(128,) → forward (64) + backward (64)
Memory Mechanism	Gates: forget, input, output + cell state
Key Strength	Learns from **both past and future** context
Training Algorithm	Backpropagation Through Time (BPTT) with **Adam optimizer**
Why in your model?	Captures **sequence meaning**, especially over longer word distances

---

Embedding Layer

🔍 Embedding Layer Parameters

Parameter	Meaning

`input_dim=5000`	Only the top 5000 most frequent words are used. Each gets a unique ID.
`output_dim=128`	Each word will be mapped to a 128-dimensional dense vector.
`input_length=200`	Input sequences are padded/truncated to length 200.

Concept	Details
Layer Used	`Embedding(input_dim=5000, output_dim=128, input_length=200)`
Input	List of word indices (integers) → shape: `(200,)`
Output	Dense word vector sequence → shape: `(200, 128)`
Learnable?	✅ Yes — weights are learned via backpropagation
Purpose	Map discrete words to continuous space that encodes semantics
Role in Model	First layer that converts tokenized emails into usable numeric format

---

Regularization and Optimization

Technique	Your Code	Purpose	Key Effect
Dropout	`Dropout(0.6)`	Prevent overfitting	Randomly drops 60% of neurons
L2 Regularization	`kernel_regularizer=l2(0.01)`	Keep weights small	Penalizes large weights
Optimizer	`Adam(learning_rate=0.001)`	Minimize loss with adaptive updates	Fast, stable training