Model Structure and Parameters Used in Study for the Deep Learning Models:

CNN

1. First is the embedding layer of the model. The embedding layer is used to convert the input sequences (integer-encoded words) into dense vectors of fixed size. The 5000 parameter specifies the vocabulary size, 128 is the dimensionality of the embedding space, and 100 is the length of input sequences.
2. Then we add a 1D convolutional layer to the model. The convolutional layer is used to apply a set of filters to the input and extract local features. The 64 parameter represents the number of filters, and 5 is the size of each filter. The activation function used is ReLU (Rectified Linear Unit).
3. Then we add a global max pooling layer. The global max pooling operation reduces the dimensionality of the input by taking the maximum value over each feature map. It helps to capture the most important features from the previous layer.
4. Then we add a dense layer to the model. The dense layer is a fully connected layer where each neuron is connected to every neuron in the previous layer. The 2 parameter specifies the number of neurons in the layer, and the activation function used is sigmoid. Since the problem seems to be a binary classification task, the output layer has 2 neurons, each representing the probability of belonging to one of the classes.
5. Then we compile the model, configuring the learning process. The optimizer used is Adam, a popular optimization algorithm. The loss function is set to binary crossentropy, which is commonly used for binary classification problems. The metric specified is accuracy, which will be used to evaluate the model's performance.

LSTM

1. First is the embedding layer of the model. The embedding layer is used to convert the input sequences (integer-encoded words) into dense vectors of fixed size. The 5000 parameter specifies the vocabulary size, 128 is the dimensionality of the embedding space, and 100 is the length of input sequences.
2. Then we add an LSTM (Long Short-Term Memory) layer to the model. LSTM is a type of recurrent neural network (RNN) that can capture long-term dependencies in sequential data. The 64 parameter represents the number of LSTM units (or cells) in the layer. The dropout parameter specifies the dropout rate, which helps prevent overfitting by randomly disabling a portion of the LSTM units during training. We add a recurrentdropout parameter that specifies the dropout rate for the recurrent connections within the LSTM units.
3. Then we add a dense output layer to the model, similar to the previous example. It has 2 neurons, representing the probability of belonging to each class. The activation function used is sigmoid.
4. Then we compile the model, configuring the learning process. The optimizer used is Adam, a popular optimization algorithm. The loss function is set to binary crossentropy, which is commonly used for binary classification problems. The metric specified is accuracy, which will be used to evaluate the model's performance.

BI-LSTM

1. First is the embedding layer of the model. The embedding layer is used to convert the input sequences (integer-encoded words) into dense vectors of fixed size. The 5000 parameter specifies the vocabulary size, 128 is the dimensionality of the embedding space, and 100 is the length of input sequences.
2. Then we add a Bidirectional LSTM layer to the model. A Bidirectional LSTM processes the input sequence in both directions, incorporating information from past and future timesteps. The 64 parameter represents the number of LSTM units (or cells) in each direction. The dropout parameter specifies the dropout rate, which helps prevent overfitting by randomly disabling a portion of the LSTM units during training. Then we add a recurrentdropout parameter that specifies the dropout rate for the recurrent connections within the LSTM units.
3. Then we add a dense output layer to the model, similar to the previous examples. It has 2 neurons, representing the probability of belonging to each class. The activation function used is sigmoid.
4. Then we compile the model, configuring the learning process. The optimizer used is Adam, a popular optimization algorithm. The loss function is set to binary crossentropy, which is commonly used for binary classification problems. The metric specified is accuracy, which will be used to evaluate the model's performance.

BERT

1. The BERT model for sequence classification consists of a BERT encoder with 12 layers, incorporating self-attention and feed-forward neural network layers. It receives input embeddings for words, positions, and token types, while utilizing layer normalization and dropout for regularization.
2. The BertEmbeddings module manages the word embeddings, position embeddings, and token-type embeddings. The word-embeddings embedding layer has a vocabulary size of 30522, producing embeddings of size 768. Both the position-embeddings and token-type-embeddings layers also generate embeddings of size 768. Layer normalization and dropout are applied to the embeddings.
3. The BertEncoder contains a list of 12 BertLayer modules, each representing a single encoder layer. Each BertLayer consists of a BertSelfAttention module for selfattention computations, a BertIntermediate module with a dense layer and GELU activation function, and a BertOutput module with a dense layer for the final output. The self-attention module applies linear transformations to the input and incorporates dropout. The intermediate module utilizes a dense layer with 3072 output features and the GELU activation function. The output module employs a dense layer with 768 output features, incorporating layer normalization and dropout.
4. The BertPooler module comprises a dense layer with 768 input features and 768 output features, followed by a hyperbolic tangent activation function.
5. The BertForSequenceClassification model includes a dropout layer with a dropout probability of 0.1. The final classification is performed by a linear layer with 768 input features and 2 output features for binary classification.

RO-BERTa

1. The Roberta For SequenceClassification model is based on the RoBERTa architecture, designed for sequence classification tasks. It utilizes a RobertaEncoder, which consists of 12 RobertaLayers. Each layer includes a self-attention mechanism and feed-forward neural network layers.
2. The input to the model includes word embeddings, position embeddings, and token type embeddings. The word-embeddings layer has a vocabulary size of 50265, generating embeddings of size 768. The position-embeddings and token-type-embeddings layers also produce embeddings of size 768. Layer normalization and dropout are applied to the embeddings for regularization.
3. The RobertaEncoder contains 12 RobertaLayers, each comprising a RobertaSelfAttention module for self-attention computations, a RobertaIntermediate module with a dense layer and GELU activation function, and a RobertaOutput module with a dense layer for the final output. The self-attention module performs linear transformations on the input, applying dropout for regularization. The intermediate module includes a dense layer with 3072 output features and the GELU activation function. The output module consists of a dense layer with 768 output features, applying layer normalization and dropout.
4. The RobertaForSequenceClassification model also includes a classifier called RobertaClassificationHead. This classifier consists of a dense layer with 768 input features and 768 output features, followed by dropout. The final classification is performed by a linear layer with 768 input features and 2 output features, suitable for binary classification tasks.
5. Overall, the model consists of 12 encoder layers, embedding layers, attention mechanisms, feed-forward layers, layer normalization, dropout, and a linear classifier. It has a total of 125 million trainable parameters, making it a powerful model for sequence classification.