```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, Dense, Dropout, Conv1D, GlobalMaxPooling1D, Attention


df = pd.read_csv('train.csv')  # Use the correct name of your CSV file here
df.head()  # Display the first few rows of the dataset
```

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

```
# Extract features and labels
X = df['comment_text']
y = df[['toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate']].values

# Tokenize and pad sequences
tokenizer = Tokenizer(num_words=20000, oov_token='<OOV>')
tokenizer.fit_on_texts(X)
X_seq = tokenizer.texts_to_sequences(X)
X_padded = pad_sequences(X_seq, maxlen=200)

# Split dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_padded, y, test_size=0.2, random_state=42)



# Input layer
input_layer = Input(shape=(200,))

# Embedding layer
embedding_layer = Embedding(input_dim=20000, output_dim=128, input_length=200)(input_layer)

# Convolutional Layer
conv_layer = Conv1D(128, 5, activation='relu')(embedding_layer)

# Attention Layer
attention_layer = Attention()([conv_layer, conv_layer])

# Global Max Pooling
pooling_layer = GlobalMaxPooling1D()(attention_layer)

# Fully connected layers
dense_layer = Dense(128, activation='relu')(pooling_layer)
dropout_layer = Dropout(0.5)(dense_layer)
output_layer = Dense(6, activation='sigmoid')(dropout_layer)  # 6 output classes for multi-label classification

# Create model
hybrid_attention_model = Model(inputs=input_layer, outputs=output_layer)

hybrid_attention_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
hybrid_attention_model.summary()  # Display the model summary
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. J
  warnings.warn(
```

Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 200)	0	-
embedding (Embedding)	(None, 200, 128)	2,560,000	input_layer[0][0]
conv1d (Conv1D)	(None, 196, 128)	82,048	embedding[0][0]
attention (Attention)	(None, 196, 128)	0	conv1d[0][0], conv1d[0][0]
global_max_pooling1d (GlobalMaxPooling1D)	(None, 128)	0	attention[0][0]
dense (Dense)	(None, 128)	16,512	global_max_pooling1d[…
dropout (Dropout)	(None, 128)	0	dense[0][0]
dense_1 (Dense)	(None, 6)	774	dropout[0][0]

Total params: 2,659,334 (10.14 MB)
Trainable params: 2,659,334 (10.14 MB)

```
# Train Hybrid Attention model
history_attention = hybrid_attention_model.fit(X_train, y_train, epochs=5, batch_size=64, validation_split=0.2)
```

```
Epoch 1/5
1596/1596 ━━━━━━━━━━━━━━━━━━━━ 495s 309ms/step - accuracy: 0.7447 - loss: 0.1390 - val_accuracy: 0.9943 - val_loss: 0.0541
Epoch 2/5
1596/1596 ━━━━━━━━━━━━━━━━━━━━ 494s 310ms/step - accuracy: 0.9851 - loss: 0.0511 - val_accuracy: 0.9943 - val_loss: 0.0522
Epoch 3/5
1596/1596 ━━━━━━━━━━━━━━━━━━━━ 480s 301ms/step - accuracy: 0.9879 - loss: 0.0422 - val_accuracy: 0.9939 - val_loss: 0.0519
Epoch 4/5
1596/1596 ━━━━━━━━━━━━━━━━━━━━ 514s 309ms/step - accuracy: 0.9583 - loss: 0.0354 - val_accuracy: 0.9941 - val_loss: 0.0580
Epoch 5/5
1596/1596 ━━━━━━━━━━━━━━━━━━━━ 498s 306ms/step - accuracy: 0.9053 - loss: 0.0294 - val_accuracy: 0.9897 - val_loss: 0.0600
```

```
# Evaluate Hybrid Attention model
attention_loss, attention_acc = hybrid_attention_model.evaluate(X_test, y_test)
print(f"Hybrid Attention Model Accuracy: {attention_acc:.4f}")
```

```
998/998 ━━━━━━━━━━━━━━━━━━━━ 45s 45ms/step - accuracy: 0.9887 - loss: 0.0561
Hybrid Attention Model Accuracy: 0.9888
```

```
import matplotlib.pyplot as plt

# Plot training & validation accuracy values
plt.plot(history_attention.history['accuracy'])
plt.plot(history_attention.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.plot(history_attention.history['loss'])
plt.plot(history_attention.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```