cnn

April 21, 2025

[1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import KFold
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Conv1D, GlobalMaxPooling1D, Dense
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import load_model
```

[3]:
```
# 1. Load your data from the CSV file
df = pd.read_csv('emails.csv')
df.sample(5)
```

[3]:

	text	spam
988	Subject: returned mail : see transcript for de…	1
2497	Subject: re : weather and energy price data m…	0
4007	Subject: re : ebs var transaction policy i ha…	0
2006	Subject: re : executive program on credit risk…	0
2231	Subject: re : check vince , ? oh . ? i sent…	0

[5]:
```
X = df['text'].astype(str).values # Email text
y = df['spam'].values  # Spam (1) or not spam (0)
```

[7]:
```
max_words = 10000  # Maximum number of words to keep
tokenizer = Tokenizer(num_words=max_words, oov_token="<unk>") # <unk> for␣
 ↪unknown words

# Fit tokenizer on the text
tokenizer.fit_on_texts(X)

# Convert text to sequences of integers
X = tokenizer.texts_to_sequences(X)
```

[9]:
```
# 3. Padding
max_len = 200  # Maximum sequence length
X = pad_sequences(X, maxlen=max_len)
```

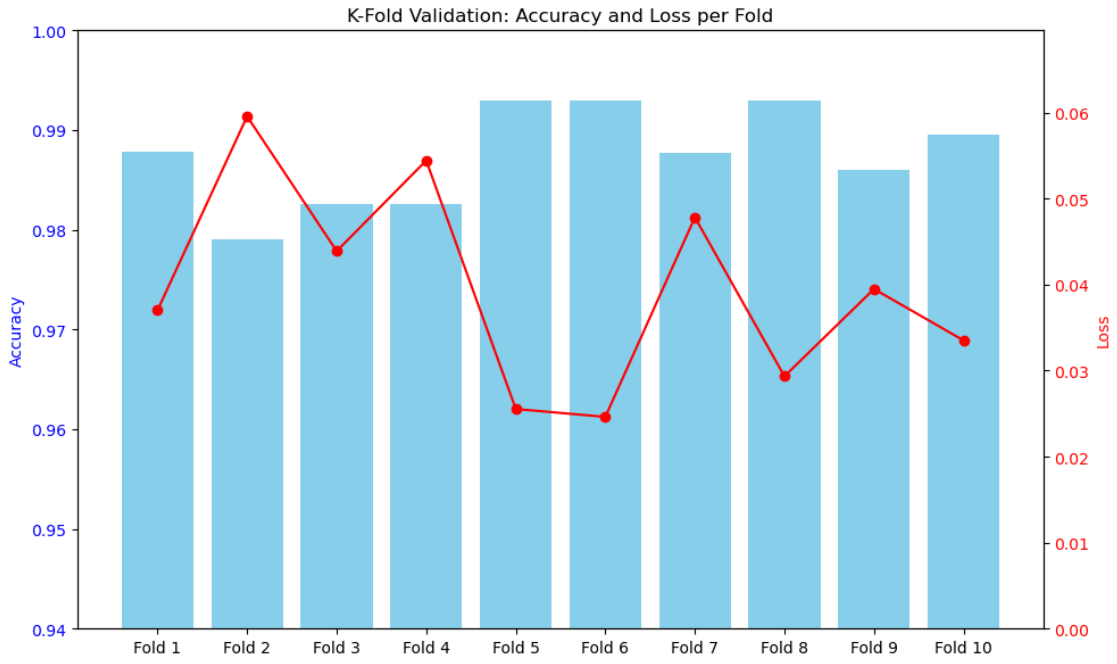
[11]:
```
# 4. Define CNN model
def create_cnn_model(vocab_size, max_len):
    model = Sequential()
    model.add(Embedding(vocab_size, 128, input_length=max_len))
    model.add(Conv1D(128, kernel_size=5, activation='relu'))
    model.add(GlobalMaxPooling1D())
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])
    return model
```

[26]:
```
# 5. K-Fold Cross-Validation
kf = KFold(n_splits=10, shuffle=True, random_state=42)
losses = []
accuracies = []
fold_no = 1
for train_index, test_index in kf.split(X, y):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Get the vocabulary size
    vocab_size = len(tokenizer.word_index) + 1  # +1 for the padding token

    # Create and train the CNN model
    cnn_model = create_cnn_model(vocab_size, max_len)
    cnn_model.fit(X_train, y_train, epochs=2, batch_size=32, verbose=0)

    # Evaluate the model
    loss, accuracy = cnn_model.evaluate(X_test, y_test, verbose=0)
    losses.append(loss)
    accuracies.append(accuracy)
    print(f"Fold {fold_no} - Loss: {loss:.4f}, Accuracy: {accuracy:.4f}")
    fold_no += 1
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\keras\src\layers\core\embedding.py:90:
UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
```

```
Fold 1 - Loss: 0.0370, Accuracy: 0.9878
Fold 2 - Loss: 0.0596, Accuracy: 0.9791
Fold 3 - Loss: 0.0439, Accuracy: 0.9825
Fold 4 - Loss: 0.0544, Accuracy: 0.9825
Fold 5 - Loss: 0.0255, Accuracy: 0.9930
Fold 6 - Loss: 0.0246, Accuracy: 0.9930
Fold 7 - Loss: 0.0478, Accuracy: 0.9878
Fold 8 - Loss: 0.0293, Accuracy: 0.9930
Fold 9 - Loss: 0.0395, Accuracy: 0.9860
Fold 10 - Loss: 0.0335, Accuracy: 0.9895
```

[30]:
```
import matplotlib.pyplot as plt
# Fold labels
folds = [f"Fold {i}" for i in range(1, 11)]
fig, ax1 = plt.subplots(figsize=(10, 6))
# Bar plot for accuracy
ax1.bar(folds, accuracies, color='skyblue', label='Accuracy')
ax1.set_ylabel('Accuracy', color='blue')
ax1.set_ylim(0.94, 1.0)
ax1.tick_params(axis='y', labelcolor='blue')
# Line plot for loss
ax2 = ax1.twinx()
ax2.plot(folds, losses, color='red', marker='o', label='Loss')
ax2.set_ylabel('Loss', color='red')
ax2.set_ylim(0, max(losses) + 0.01)
ax2.tick_params(axis='y', labelcolor='red')
plt.title('K-Fold Validation: Accuracy and Loss per Fold')
plt.tight_layout()
plt.show()
```



[28]:
```
cnn_model.save('cnn_model.keras')
```

[30]:
```
model = load_model('cnn_model.keras')
import shutil
shutil.copy('cnn_model.keras', 'cnn_model.mds')
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\keras\src\saving\saving_lib.py:757:
```

```
UserWarning: Skipping variable loading for optimizer 'rmsprop', because it has 7
variables whereas the saved optimizer has 12 variables.
  saveable.load_own_variables(weights_store.get(inner_path))
```

[30]:
```
'cnn_model.mds'
```

[34]:
```
import pickle
with open('tokenizer.pickle', 'wb') as handle:
    pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

[15]:
```
# 6. Define a function to predict spam or not spam
def predict_email(email_text, model, tokenizer, max_len=200):
    sequence = tokenizer.texts_to_sequences([email_text])
    padded_sequence = pad_sequences(sequence, maxlen=max_len)
    prediction = model.predict(padded_sequence)[0][0]
    return "Spam" if prediction >= 0.5 else "Not Spam"
```

[17]:
```
# 7. Take input from user
user_email = input("Enter the email text: ")
result = predict_email(user_email, cnn_model, tokenizer)
print("Prediction:", result)
```

```
Enter the email text:  Subject: save your money by getting an oem software !
need in software for your pc ? just visit our site , we might have what you need
. . .  best regards ,  alyssa
1/1             0s 201ms/step
Prediction: Spam
```

[ ]:
```

```