Program 5:

Design and implement a deep learning network for classification of textual documents.

```
from tensorflow.keras.datasets import reuters
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Dense, LSTM, GlobalAveragePooling1D
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Load Reuters dataset
(vocab_train, y_train), (vocab_test, y_test) = reuters.load_data(num_words=10000)
x_train = pad_sequences(vocab_train, maxlen=200)
x_test = pad_sequences(vocab_test, maxlen=200)

# Build model
model = Sequential([
    Embedding(input_dim=10000, output_dim=64, input_length=200),
    LSTM(64),
    Dense(46, activation='softmax')  # 46 topic classes
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=3, batch_size=128, validation_data=(x_test, y_test))
```

Program 8:

```
# Dataset: IMDB Movie Reviews
import tensorflow as tf
from tensorflow.keras import layers, models, datasets, preprocessing
import matplotlib.pyplot as plt

# 1. Load the IMDB dataset
(x_train, y_train), (x_test, y_test) = datasets.imdb.load_data(num_words=5000)

# 2. Pad all sequences to the same length (200 words)
x_train = preprocessing.sequence.pad_sequences(x_train, maxlen=200)
x_test = preprocessing.sequence.pad_sequences(x_test, maxlen=200)

# 3. Build a simple Deep Learning model
model = models.Sequential([
    layers.Embedding(input_dim=5000, output_dim=64, input_length=200),
    layers.GlobalAveragePooling1D(),
    layers.Dense(32, activation='relu'),
    layers.Dense(1, activation='sigmoid')  # 0 = negative, 1 = positive
])

# 4. Compile the model
model.compile(optimizer='adam',
        loss='binary_crossentropy',
        metrics=['accuracy'])

# 5. Train the model
history = model.fit(
    x_train, y_train,
    epochs=5,
    batch_size=128,
    validation_data=(x_test, y_test)
)

# 6. Evaluate the model on test data
loss, accuracy = model.evaluate(x_test, y_test, verbose=2)
print(f"\nTest Accuracy: {accuracy:.4f}")

# 7. Plot Training Accuracy & Validation Accuracy
plt.figure(figsize=(8,4))
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# 8. Plot Training Loss & Validation Loss
plt.figure(figsize=(8,4))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
Epoch 5/5
196/196 [==============================] - 2s 9ms/step - loss: 0.2843 - accuracy: 0.8885 - val_loss: 0.3144 - val_accuracy: 0.8701

782/782 - 3s - loss: 0.3132 - accuracy: 0.8710
Test Accuracy: 0.8710
```

Pgm 8 :

```
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Load IMDB dataset
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=10000)
x_train = pad_sequences(x_train, maxlen=200)
x_test = pad_sequences(x_test, maxlen=200)

# Build model
model = Sequential([
    Embedding(input_dim=10000, output_dim=64, input_length=200),
    LSTM(64),
    Dense(1, activation='sigmoid')  # Binary output
])
```

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

model.fit(x_train, y_train, epochs=3, batch_size=128, validation_data=(x_test, y_test))
```