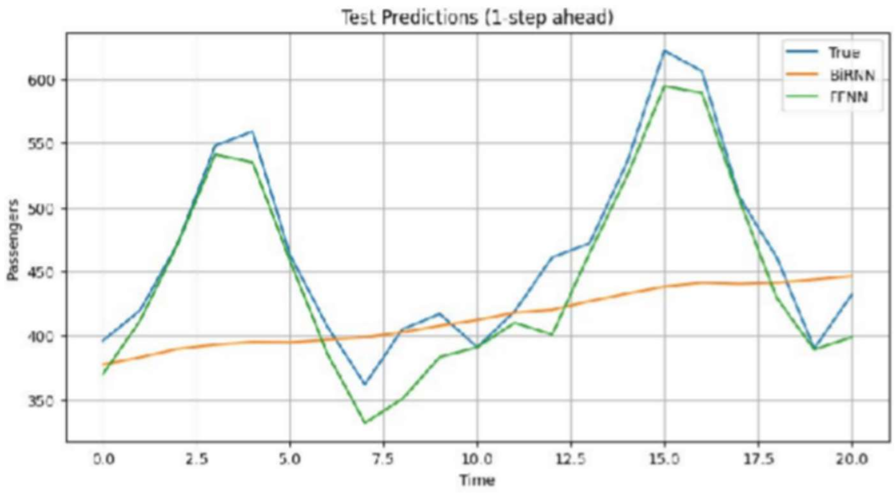
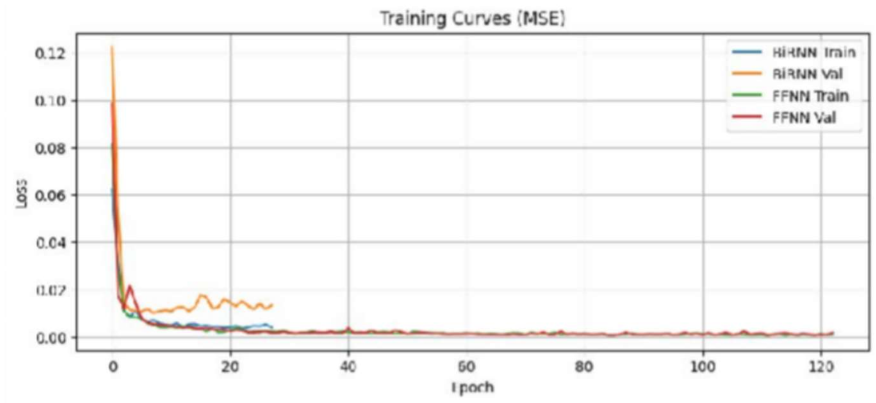
```
plt.title("Test Predictions (1-step ahead)"); plt.xlabel("Time"); plt.ylabel("Passengers");
plt.legend(); plt.grid(); plt.show()


plt.figure(figsize=(10,4)); plt.plot(hist_birnn.history["loss"],label="BiRNN Train");
plt.plot(hist_birnn.history["val_loss"],label="BiRNN Val")

plt.plot(hist_ffnn.history["loss"],label="FFNN Train");
plt.plot(hist_ffnn.history["val_loss"],label="FFNN Val")

plt.title("Training Curves (MSE)"); plt.xlabel("Epoch"); plt.ylabel("Loss"); plt.legend();
plt.grid(); plt.show()


winner = "BiRNN" if metrics_birnn["RMSE"]<metrics_ffnn["RMSE"] else "FFNN"

print(f"\nWinner by RMSE: {winner}")
```

OUTPUT:





RESULT: The FFNN achieved lower error metrics and outperformed the Bidirectional RNN for sequence prediction tasks.

EXP NO: 07 DATE: 20/09/2025	**CAPTION GENERATION USING RNN+CNN**

AIM: To build a deep recurrent neural network (RNN) that generates image captions by combining a Convolutional Neural Network (CNN) for image feature extraction with an RNN for sequence generation using the MS COCO (or Flickr8k) dataset.

ALGORITHM:

- Import TensorFlow, Keras, and supporting libraries.
- Load and preprocess the caption dataset, adding start and end tokens to each caption.
- Use a pre-trained CNN model (InceptionV3) to extract feature vectors from images.
- Tokenize and pad captions, converting text to numerical sequences.
- Create input-output pairs combining image features and partial text sequences for training.
- Define a multimodal model combining CNN features and RNN outputs: CNN output passes through a dense layer, Captions are embedded and processed through an LSTM, their outputs are merged and passed to a softmax layer for word prediction.
- Train the model using sparse categorical cross-entropy loss.
- Generate captions by iteratively predicting the next word until the end token is reached.
- Display the image with the generated caption for evaluation.

CODE:

```
import tensorflow as tf

from tensorflow.keras.applications import InceptionV3

from tensorflow.keras.applications.inception_v3 import preprocess_input

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad_sequences

from tensorflow.keras.layers import Input, Embedding, LSTM, Dense, Add

from tensorflow.keras.models import Model

import numpy as np, os, pandas as pd

from PIL import Image

from tqdm import tqdm

import matplotlib.pyplot as plt
```

```
# GPU configuration
gpus = tf.config.list_physical_devices('GPU')
if gpus:
    try:
        for gpu in gpus:
            tf.config.experimental.set_memory_growth(gpu, True)
        tf.keras.mixed_precision.set_global_policy('mixed_float16')
    except RuntimeError as e:
        print(f"GPU configuration error: {e}")

print(f"TensorFlow version: {tf.__version__}")

# Paths
IMG_DIR = "Images"
CAP_FILE = "captions.txt"

# Load captions
df = pd.read_csv(CAP_FILE)
df['caption'] = df['caption'].apply(lambda x: 'startseq ' + x.lower() + ' endseq')

# Subset for demo
df = df.groupby('image').head(1).sample(2000, random_state=42)

# CNN feature extractor
cnn = InceptionV3(weights='imagenet', include_top=False, pooling='avg')

def extract_feat(img_path):
    img = Image.open(img_path).convert('RGB').resize((299,299))
    x = np.expand_dims(preprocess_input(np.array(img)), 0)
    return cnn.predict(x, verbose=0)[0]

# Extract features
features, captions = [], []
for img, cap in tqdm(zip(df['image'], df['caption']), total=len(df)):
    path = os.path.join(IMG_DIR, img)
    if os.path.exists(path):
        features.append(extract_feat(path))
        captions.append(cap)
features = np.array(features)

# Tokenize captions
tok = Tokenizer(num_words=5000, oov_token='unk')
tok.fit_on_texts(captions)
seqs = tok.texts_to_sequences(captions)
maxlen = max(len(s) for s in seqs)
vocab = len(tok.word_index) + 1

# Prepare training data
X1, X2, y = [], [], []
for f, s in zip(features, seqs):
    for i in range(1, len(s)):
        in_seq, out = s[:i], s[i]
        X1.append(f)
        X2.append(pad_sequences([in_seq], maxlen=maxlen)[0])
        y.append(out)
```