In [2]:
```
import tensorflow as tf
from tensorflow.keras import layers, models, datasets
import matplotlib.pyplot as plt
import numpy as np
```

In [3]:
```
# Load CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = datasets.cifar10.load_data()

# Normalize pixel values to between 0 and 1
x_train, x_test = x_train / 255.0, x_test / 255.0

# Convert labels to one-hot encoding
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
```

In [4]:
```
def build_ann_model():
    model = models.Sequential([
        layers.Flatten(input_shape=(32, 32, 3)),
        layers.Dense(128, activation='relu'),
        layers.Dense(64, activation='relu'),
        layers.Dense(10, activation='softmax')
    ])
    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

In [5]:

```
def build_cnn_model():
    model = models.Sequential([
        layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.Flatten(),
        layers.Dense(64, activation='relu'),
        layers.Dense(10, activation='softmax')
    ])
    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

In [6]:

```
# ANN Model
ann_model = build_ann_model()
history_ann = ann_model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))

# CNN Model
cnn_model = build_cnn_model()
history_cnn = cnn_model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input_shape
`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer i
n the model instead.
  super().__init__(**kwargs)
Epoch 1/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 9s 4ms/step - accuracy: 0.2757 - loss: 1.9957 - val_accuracy: 0.3713 - val_loss: 1.7350
Epoch 2/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.3802 - loss: 1.7247 - val_accuracy: 0.3942 - val_loss: 1.6755
Epoch 3/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.4115 - loss: 1.6580 - val_accuracy: 0.4262 - val_loss: 1.6132
Epoch 4/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.4294 - loss: 1.5973 - val_accuracy: 0.4171 - val_loss: 1.6320
Epoch 5/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.4442 - loss: 1.5613 - val_accuracy: 0.4506 - val_loss: 1.5480
Epoch 6/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.4495 - loss: 1.5312 - val_accuracy: 0.4490 - val_loss: 1.5459
Epoch 7/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.4591 - loss: 1.5194 - val_accuracy: 0.4314 - val_loss: 1.5842
Epoch 8/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.4721 - loss: 1.4873 - val_accuracy: 0.4538 - val_loss: 1.5301
Epoch 9/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.4748 - loss: 1.4733 - val_accuracy: 0.4648 - val_loss: 1.4992
Epoch 10/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.4735 - loss: 1.4712 - val_accuracy: 0.4585 - val_loss: 1.5134
```

```
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.4735 - loss: 1.4712 - val_accuracy: 0.4585 - val_loss: 1.5134
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input
_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first l
ayer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 12s 5ms/step - accuracy: 0.3539 - loss: 1.7382 - val_accuracy: 0.5786 - val_loss: 1.2016
Epoch 2/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.5849 - loss: 1.1687 - val_accuracy: 0.6099 - val_loss: 1.1143
Epoch 3/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.6551 - loss: 0.9861 - val_accuracy: 0.6631 - val_loss: 0.9657
Epoch 4/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.6944 - loss: 0.8765 - val_accuracy: 0.6896 - val_loss: 0.8845
Epoch 5/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - accuracy: 0.7192 - loss: 0.7981 - val_accuracy: 0.6903 - val_loss: 0.8799
Epoch 6/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.7446 - loss: 0.7338 - val_accuracy: 0.6810 - val_loss: 0.9613
Epoch 7/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7608 - loss: 0.6840 - val_accuracy: 0.7054 - val_loss: 0.8423
Epoch 8/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.7769 - loss: 0.6344 - val_accuracy: 0.7123 - val_loss: 0.8545
Epoch 9/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.7894 - loss: 0.5949 - val_accuracy: 0.7116 - val_loss: 0.8689
Epoch 10/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.8040 - loss: 0.5582 - val_accuracy: 0.7003 - val_loss: 0.9008
```

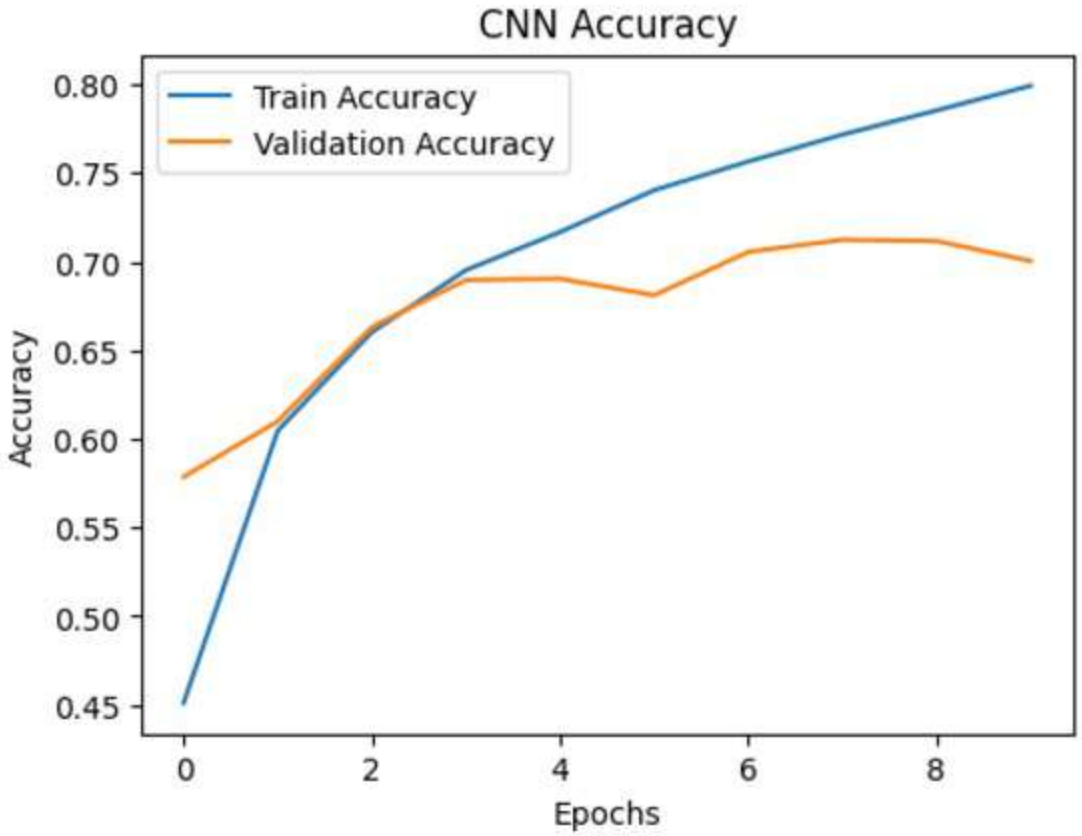
In [7]:

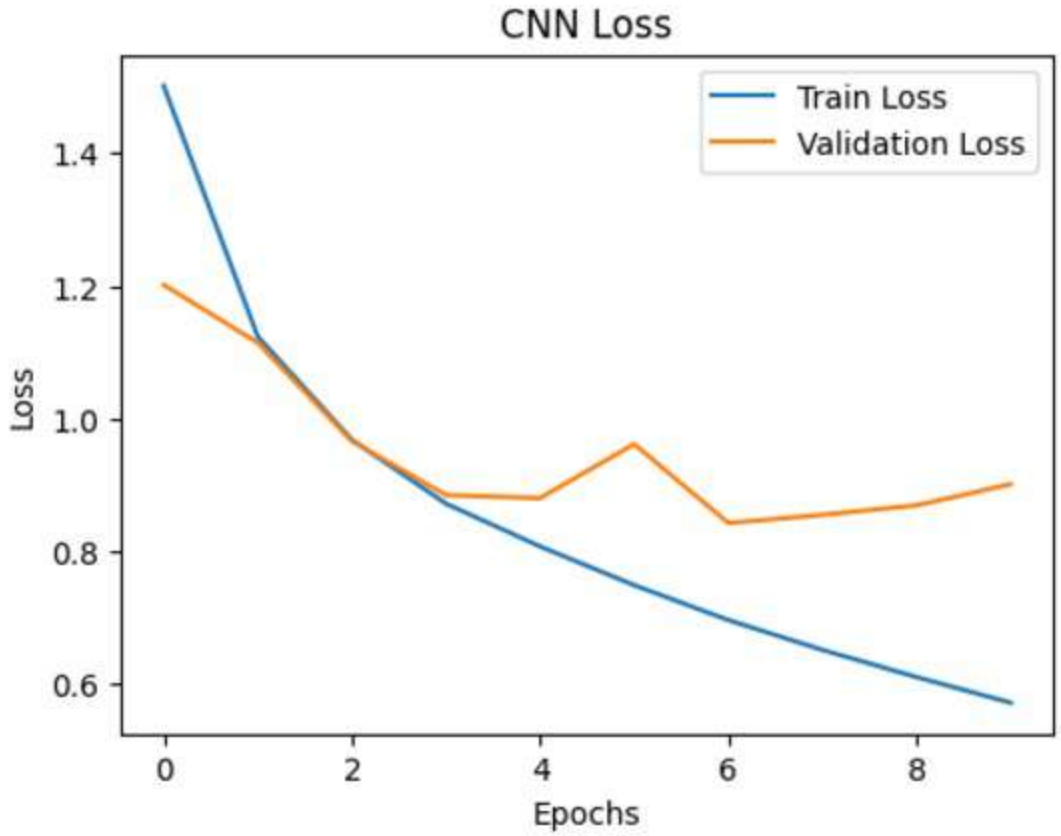
```python
# ANN Evaluation
ann_test_loss, ann_test_acc = ann_model.evaluate(x_test, y_test)
print(f"ANN Test Accuracy: {ann_test_acc}")

# CNN Evaluation
cnn_test_loss, cnn_test_acc = cnn_model.evaluate(x_test, y_test)
print(f"CNN Test Accuracy: {cnn_test_acc}")
```

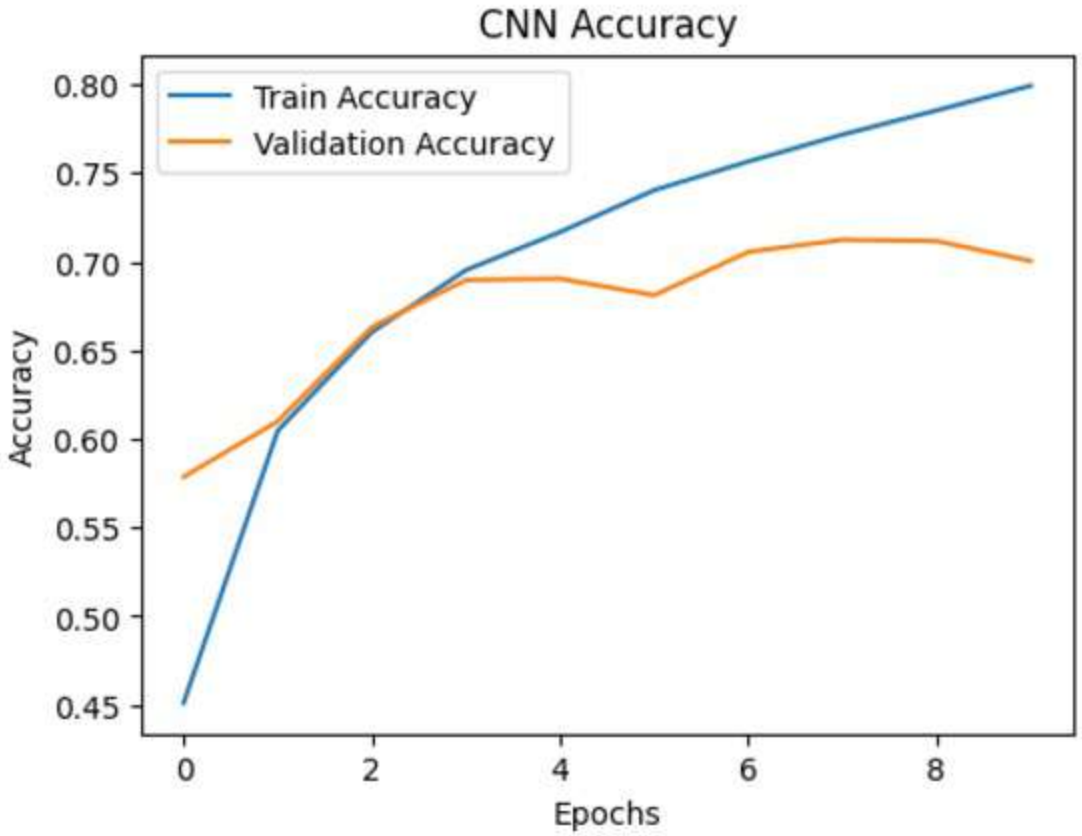
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.4573 - loss: 1.5025
ANN Test Accuracy: 0.4584999978542328
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.7076 - loss: 0.8835
CNN Test Accuracy: 0.7002999782562256
```

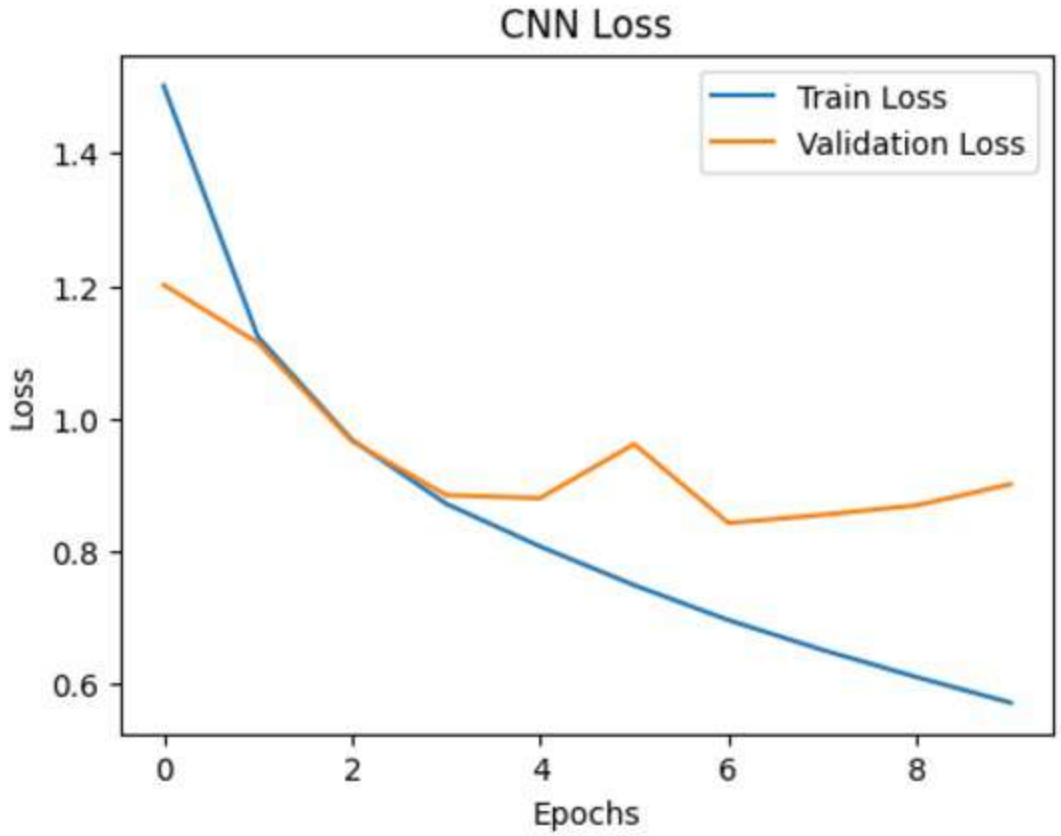
In [8]:

```
def plot_history(history, title):
    plt.figure(figsize=(12, 4))

    # Plot accuracy
    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'], label='Train Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.title(f'{title} Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()

    # Plot loss
    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'], label='Train Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.title(f'{title} Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()

    plt.show()

# Plot for ANN
plot_history(history_ann, "ANN")

# Plot for CNN
plot_history(history_cnn, "CNN")
```


CNN Accuracy
Train Accuracy
Validation Accuracy
Accuracy
Epochs
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0
2
4
6
8


CNN Loss
Train Loss
Validation Loss
Loss
Epochs
1.4
1.2
1.0
0.8
0.6
0
2
4
6
8


CNN Accuracy
Train Accuracy
Validation Accuracy
Accuracy
Epochs
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0
2
4
6
8


CNN Loss
Train Loss
Validation Loss
Loss
Epochs
1.4
1.2
1.0
0.8
0.6
0
2
4
6
8