In [1]:

```python
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input,Conv2D, MaxPooling2D, Flatten, Der
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix

# Load CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Normalize images (scaling pixel values between 0 and 1)
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

# Convert labels to categorical (one-hot encoding)
y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)

# Define class names
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']
#verify datasetshape
print(f"Training data shape:{x_train.shape}")
print(f"Test data shape:{x_test.shape}")

# Build CNN model
model = Sequential([
    Input(shape=(32,32,3)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')
])
model.summary()

# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
print(x_train.shape,y_train.shape)
print(x_test.shape,y_test.shape)

# Train the model
history = model.fit(x_train, y_train, epochs=10, batch_size=64, validation_c

# Evaluate the model
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
```

Loading [MathJax]/extensions/Safe.js

```python
print(f"Test Accuracy: {test_acc:.4f}")

# Predictions
y_pred = model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true_classes = np.argmax(y_test, axis=1)

# Classification Report
print("Classification Report:")
print(classification_report(y_true_classes, y_pred_classes, target_names=cla

# Confusion Matrix
conf_matrix = confusion_matrix(y_true_classes, y_pred_classes)
print("Confusion Matrix:\n", conf_matrix)

# Plot training loss and accuracy
plt.figure(figsize=(12, 5))

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

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

plt.show()

# Predict sample images
def plot_sample_predictions():
    fig, axes = plt.subplots(3, 5, figsize=(10, 6))
    axes = axes.ravel()
    for i in range(15):
        index = np.random.randint(0, len(x_test))
        axes[i].imshow(x_test[index])
        axes[i].set_title(f"Pred: {class_names[y_pred_classes[index]]}\nTrue
        axes[i].axis('off')
    plt.tight_layout()
    plt.show()

plot_sample_predictions()
```

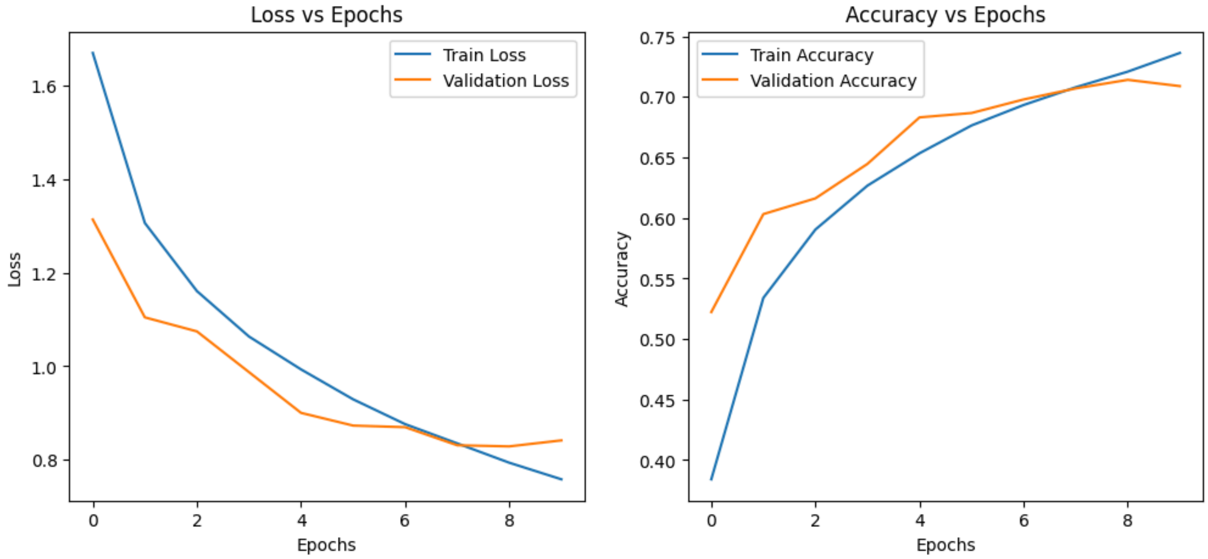
```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.g
z
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 4s 0us/step
Training data shape:(50000, 32, 32, 3)
Test data shape:(10000, 32, 32, 3)
```

```
Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                      │ (None, 30, 30, 32)          │
├──────────────────────────────────────┼─────────────────────────────┤
│ max_pooling2d (MaxPooling2D)         │ (None, 15, 15, 32)          │
├──────────────────────────────────────┼─────────────────────────────┤
│ conv2d_1 (Conv2D)                    │ (None, 13, 13, 64)          │
├──────────────────────────────────────┼─────────────────────────────┤
│ max_pooling2d_1 (MaxPooling2D)       │ (None, 6, 6, 64)            │
├──────────────────────────────────────┼─────────────────────────────┤
│ conv2d_2 (Conv2D)                    │ (None, 4, 4, 64)            │
├──────────────────────────────────────┼─────────────────────────────┤
│ flatten (Flatten)                    │ (None, 1024)                │
├──────────────────────────────────────┼─────────────────────────────┤
│ dense (Dense)                        │ (None, 128)                 │
├──────────────────────────────────────┼─────────────────────────────┤
│ dropout (Dropout)                    │ (None, 128)                 │
├──────────────────────────────────────┼─────────────────────────────┤
│ dense_1 (Dense)                      │ (None, 10)                  │
└──────────────────────────────────────┴─────────────────────────────┘

 Total params: 188,810 (737.54 KB)

 Trainable params: 188,810 (737.54 KB)

 Non-trainable params: 0 (0.00 B)
```

```
(50000, 32, 32, 3) (50000, 10)
(10000, 32, 32, 3) (10000, 10)
Epoch 1/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 64s 78ms/step - accuracy: 0.2863 - loss: 1.8959
- val_accuracy: 0.5223 - val_loss: 1.3139
Epoch 2/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 80s 76ms/step - accuracy: 0.5133 - loss: 1.3600
- val_accuracy: 0.6032 - val_loss: 1.1047
Epoch 3/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 59s 75ms/step - accuracy: 0.5843 - loss: 1.1781
- val_accuracy: 0.6162 - val_loss: 1.0747
Epoch 4/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 81s 75ms/step - accuracy: 0.6193 - loss: 1.0787
- val_accuracy: 0.6447 - val_loss: 0.9876
Epoch 5/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 59s 75ms/step - accuracy: 0.6518 - loss: 0.9963
- val_accuracy: 0.6831 - val_loss: 0.9005
Epoch 6/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 83s 77ms/step - accuracy: 0.6784 - loss: 0.9313
- val_accuracy: 0.6867 - val_loss: 0.8734
Epoch 7/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 86s 82ms/step - accuracy: 0.6922 - loss: 0.8810
- val_accuracy: 0.6979 - val_loss: 0.8699
Epoch 8/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 79s 78ms/step - accuracy: 0.7098 - loss: 0.8265
- val_accuracy: 0.7070 - val_loss: 0.8313
Epoch 9/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 80s 75ms/step - accuracy: 0.7218 - loss: 0.7913
- val_accuracy: 0.7141 - val_loss: 0.8289
Epoch 10/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 57s 73ms/step - accuracy: 0.7340 - loss: 0.7635
- val_accuracy: 0.7089 - val_loss: 0.8417
313/313 - 4s - 13ms/step - accuracy: 0.7089 - loss: 0.8417
Test Accuracy: 0.7089
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step
Classification Report:
              precision    recall  f1-score   support

    airplane       0.78      0.68      0.73      1000
  automobile       0.89      0.80      0.84      1000
        bird       0.67      0.55      0.60      1000
         cat       0.51      0.55      0.53      1000
        deer       0.63      0.63      0.63      1000
         dog       0.66      0.55      0.60      1000
        frog       0.78      0.79      0.78      1000
       horse       0.69      0.82      0.75      1000
        ship       0.72      0.89      0.80      1000
       truck       0.77      0.83      0.80      1000

    accuracy                           0.71     10000
   macro avg       0.71      0.71      0.71     10000
weighted avg       0.71      0.71      0.71     10000

Confusion Matrix:
 [[676  18  56  17  23   3   7  17 150  33]
 [ 21 797   4  18   3   1   6   7  35 108]
```

Loading [MathJax]/extensions/Safe.js

```
[ 57   2 551  76 116  53  64  46  19  16]
[ 18   7  56 545  81 145  58  41  28  21]
[ 15   2  51  71 630  13  60 131  24   3]
[  5   0  44 201  54 553  20  83  23  17]
[  6   4  22  72  47  16 794  13  15  11]
[ 12   2  25  37  43  39   4 816   7  15]
[ 31  16   8  18   2   4   3   4 894  20]
[ 21  45   6  15   1   5   7  18  49 833]]
```


Loss vs Epochs
Train Loss
Validation Loss
Loss
Epochs
Accuracy vs Epochs
Train Accuracy
Validation Accuracy
Accuracy
Epochs