In [1]:
```
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical

(x_train, y_train), (x_test, y_test) = cifar10.load_data()

x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

y_train_cat = to_categorical(y_train, 10)
y_test_cat = to_categorical(y_test, 10)
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 156s 1us/step
```

In [2]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.4),
    Dense(10, activation='softmax')
])

optimizer = Adam(learning_rate=0.0005)

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

```
C:\Users\Admin\anaconda3\Lib\site-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 layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [3]:
```
history = model.fit(x_train, y_train_cat, epochs=10, batch_size=64, validation_split=0.1)
```

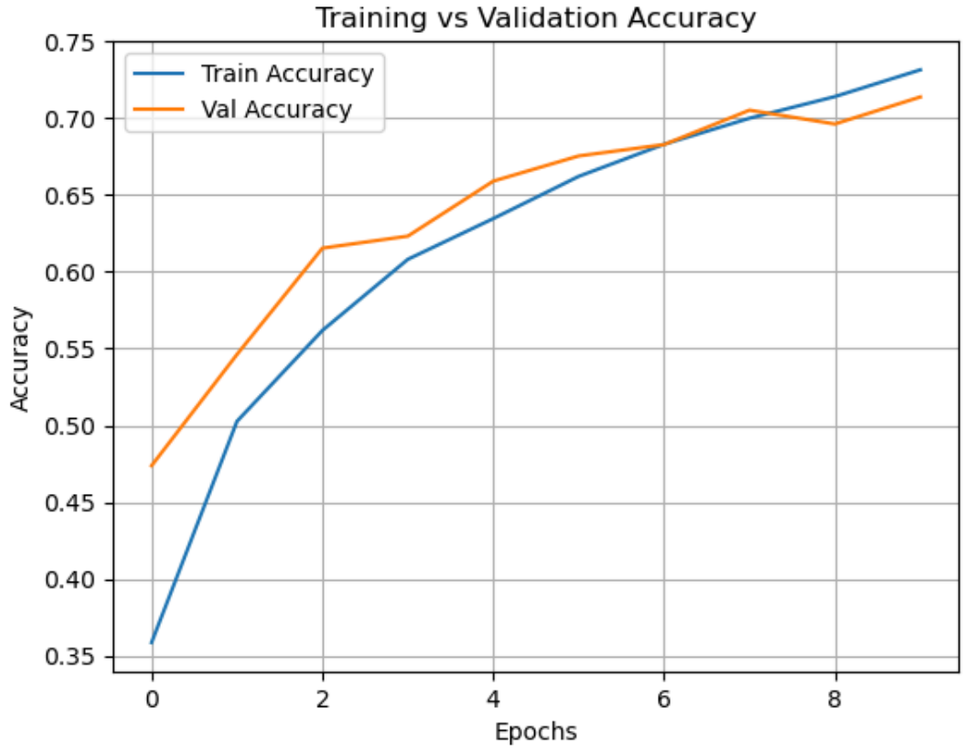
```
Epoch 1/10
704/704 ━━━━━━━━━━━━━━━━━━━━ 35s 42ms/step - accuracy: 0.2736 - loss: 1.9411 - val_accuracy: 0.4738 - val_loss: 1.4408
Epoch 2/10
704/704 ━━━━━━━━━━━━━━━━━━━━ 27s 38ms/step - accuracy: 0.4857 - loss: 1.4184 - val_accuracy: 0.5462 - val_loss: 1.2873
Epoch 3/10
704/704 ━━━━━━━━━━━━━━━━━━━━ 27s 39ms/step - accuracy: 0.5493 - loss: 1.2540 - val_accuracy: 0.6154 - val_loss: 1.0920
Epoch 4/10
704/704 ━━━━━━━━━━━━━━━━━━━━ 28s 39ms/step - accuracy: 0.6009 - loss: 1.1302 - val_accuracy: 0.6232 - val_loss: 1.0605
Epoch 5/10
704/704 ━━━━━━━━━━━━━━━━━━━━ 27s 38ms/step - accuracy: 0.6331 - loss: 1.0469 - val_accuracy: 0.6590 - val_loss: 0.9765
Epoch 6/10
704/704 ━━━━━━━━━━━━━━━━━━━━ 27s 39ms/step - accuracy: 0.6621 - loss: 0.9683 - val_accuracy: 0.6754 - val_loss: 0.9534
Epoch 7/10
704/704 ━━━━━━━━━━━━━━━━━━━━ 28s 39ms/step - accuracy: 0.6797 - loss: 0.9115 - val_accuracy: 0.6828 - val_loss: 0.8964
Epoch 8/10
704/704 ━━━━━━━━━━━━━━━━━━━━ 28s 39ms/step - accuracy: 0.6981 - loss: 0.8598 - val_accuracy: 0.7052 - val_loss: 0.8627
Epoch 9/10
704/704 ━━━━━━━━━━━━━━━━━━━━ 28s 40ms/step - accuracy: 0.7154 - loss: 0.8120 - val_accuracy: 0.6962 - val_loss: 0.8718
Epoch 10/10
704/704 ━━━━━━━━━━━━━━━━━━━━ 27s 38ms/step - accuracy: 0.7289 - loss: 0.7813 - val_accuracy: 0.7138 - val_loss: 0.8232
```

In [4]:
```python
test_loss, test_acc = model.evaluate(x_test, y_test_cat)
print(f"Test Accuracy: {test_acc:.4f}")
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - accuracy: 0.7090 - loss: 0.8377
Test Accuracy: 0.7044
```

In [5]:
```python
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.title("Training vs Validation Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.grid(True)
plt.show()
```


Training vs Validation Accuracy
Train Accuracy
Val Accuracy
Accuracy
Epochs
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0
2
4
6
8

In [ ]: