Import Required Libraries

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Load CIFAR-10 Dataset

In [2]:
```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

print("Training data shape:", x_train.shape)
print("Testing data shape:", x_test.shape)
```

```
Training data shape: (50000, 32, 32, 3)
Testing data shape: (10000, 32, 32, 3)
```

Normalize Image Data

In [3]:
```
x_train = x_train.astype('float32') / 255.0
x_test  = x_test.astype('float32') / 255.0
```

One-Hot Encode Labels

In [4]:
```
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test  = tf.keras.utils.to_categorical(y_test, 10)
```

Data Augmentation

In [5]:
```
datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True
)

datagen.fit(x_train)
```

Build CNN Model (Dropout + Dense Layers)

In [6]:
```
model = models.Sequential()
```

```
# Convolution Block 1
model.add(layers.Conv2D(32, (3,3), activation='relu', input_shape=(32,32,3)))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Dropout(0.25))

# Convolution Block 2
model.add(layers.Conv2D(64, (3,3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Dropout(0.25))

# Fully Connected Dense Network
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(128, activation='relu'))

# Output Layer
model.add(layers.Dense(10, activation='softmax'))
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_con
v.py:113: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a l
ayer. When using Sequential models, prefer using an `Input(shape)` object as th
e first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Compile the Model

In [7]:
```
model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Model Summary

In [8]:
```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
dropout (Dropout)	(None, 15, 15, 32)	0
conv2d_1 (Conv2D)	(None, 13, 13, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
dropout_1 (Dropout)	(None, 6, 6, 64)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 256)	590,080
dropout_2 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32,896
dense_2 (Dense)	(None, 10)	1,290

Total params: 643,658 (2.46 MB)

Trainable params: 643,658 (2.46 MB)

Non-trainable params: 0 (0.00 B)

Train the Model with Data Augmentation

In [9]:
```
history = model.fit(
    datagen.flow(x_train, y_train, batch_size=64),
    epochs=5,
    validation_data=(x_test, y_test)
)
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/trainers/data_adapters/py_dat
aset_adapter.py:121: UserWarning: Your `PyDataset` class should call `supe
r().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`,
`use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fi
t()`, as they will be ignored.
  self._warn_if_super_not_called()
```

```
Epoch 1/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 104s 130ms/step - accuracy: 0.2585 - loss: 1.9697
- val_accuracy: 0.4741 - val_loss: 1.4368
Epoch 2/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 101s 130ms/step - accuracy: 0.4563 - loss: 1.4958
- val_accuracy: 0.5639 - val_loss: 1.2191
Epoch 3/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 97s 125ms/step - accuracy: 0.5007 - loss: 1.3862 -
val_accuracy: 0.5947 - val_loss: 1.1651
Epoch 4/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 98s 126ms/step - accuracy: 0.5306 - loss: 1.3108 -
val_accuracy: 0.6198 - val_loss: 1.0975
Epoch 5/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 98s 125ms/step - accuracy: 0.5457 - loss: 1.2671 -
val_accuracy: 0.6430 - val_loss: 1.0164
```

Evaluate the Model

In [10]:
```python
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print("Test Accuracy:", test_accuracy)
```

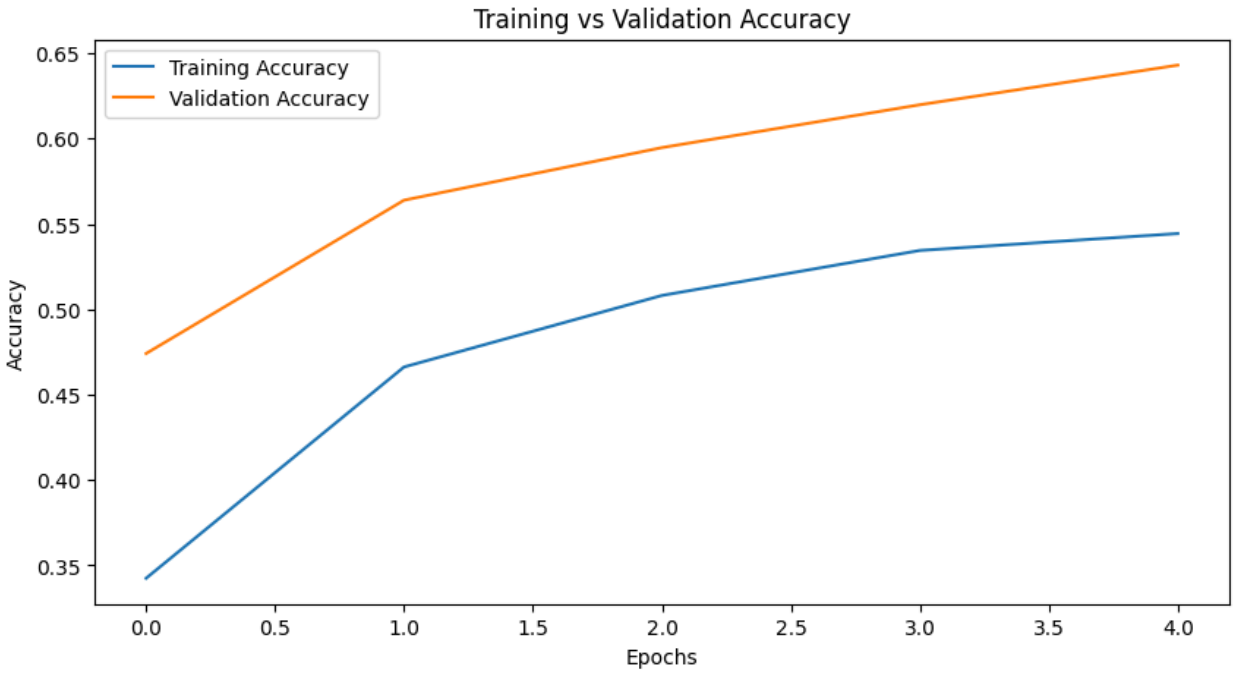
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - accuracy: 0.6470 - loss: 1.0076
Test Accuracy: 0.6430000066757202
```

Plot Accuracy & Validation Accuracy Curve

In [11]:
```python
plt.figure(figsize=(10,5))

plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')

plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Training vs Validation Accuracy')
plt.legend()
plt.show()
```

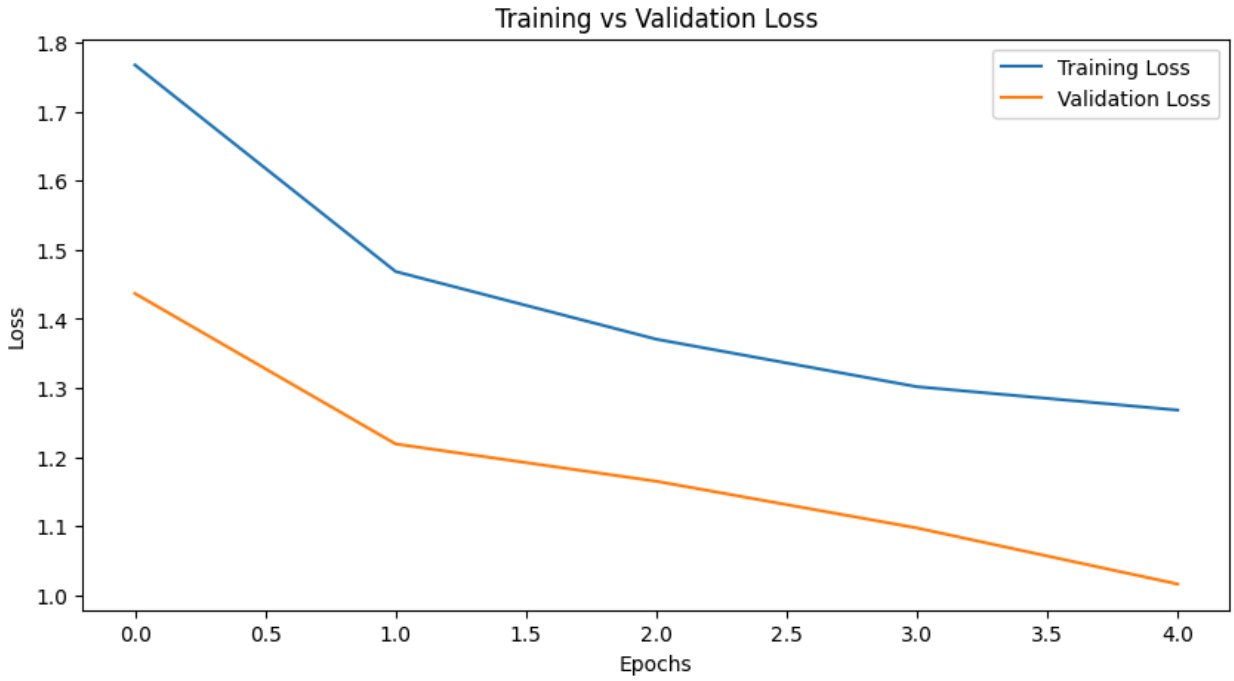


Plot Training & Validation Loss

In [12]:
```
plt.figure(figsize=(10,5))

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')

plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training vs Validation Loss')
plt.legend()
plt.show()
```


Training vs Validation Loss
Training Loss
Validation Loss
Loss
Epochs
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0