In [1]:
```python
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
```

In [2]:
```python
# Load and preprocess the CIFAR-10 dataset
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_da

# Normalize pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0
```

In [3]:
```python
model = models.Sequential([
    # Flatten the input image (from 32x32x3 to 3072)
    layers.Flatten(input_shape=(32, 32, 3)),
    # Fully connected layer with 128 neurons and ReLU activation
    layers.Dense(128, activation='relu'),
    # Fully connected layer with 64 neurons and ReLU activation
    layers.Dense(64, activation='relu'),
    # Output layer with 10 neurons (one for each class) and softmax activation
    layers.Dense(10, activation='softmax')
])
```

In [7]:
```python
# Compile the model with SGD optimizer and categorical crossentropy loss
model.compile(optimizer='sgd',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train the model with a specified number of epochs
history = model.fit(train_images, train_labels, epochs=20,
                    validation_data=(test_images, test_labels))
```

```
Epoch 1/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.4048 - accurac
y: 0.5025 - val_loss: 1.4387 - val_accuracy: 0.4872
Epoch 2/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.3899 - accurac
y: 0.5070 - val_loss: 1.5052 - val_accuracy: 0.4627
Epoch 3/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.3697 - accurac
y: 0.5159 - val_loss: 1.4510 - val_accuracy: 0.4876
Epoch 4/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.3553 - accurac
y: 0.5189 - val_loss: 1.4367 - val_accuracy: 0.4871
Epoch 5/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.3391 - accurac
y: 0.5274 - val_loss: 1.4362 - val_accuracy: 0.4921
Epoch 6/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.3242 - accurac
y: 0.5302 - val_loss: 1.4712 - val_accuracy: 0.4762
Epoch 7/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.3089 - accurac
y: 0.5357 - val_loss: 1.4599 - val_accuracy: 0.4806
Epoch 8/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.2975 - accurac
y: 0.5412 - val_loss: 1.4281 - val_accuracy: 0.4943
Epoch 9/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.2847 - accurac
y: 0.5448 - val_loss: 1.4626 - val_accuracy: 0.4795
Epoch 10/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.2764 - accurac
y: 0.5494 - val_loss: 1.4785 - val_accuracy: 0.4736
Epoch 11/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.2641 - accurac
y: 0.5502 - val_loss: 1.4020 - val_accuracy: 0.5038
Epoch 12/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.2520 - accurac
y: 0.5562 - val_loss: 1.3919 - val_accuracy: 0.5044
Epoch 13/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.2392 - accurac
y: 0.5620 - val_loss: 1.4540 - val_accuracy: 0.4839
Epoch 14/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.2311 - accurac
y: 0.5632 - val_loss: 1.3847 - val_accuracy: 0.5138
Epoch 15/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.2218 - accurac
y: 0.5679 - val_loss: 1.4646 - val_accuracy: 0.4852
Epoch 16/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.2127 - accurac
y: 0.5683 - val_loss: 1.4166 - val_accuracy: 0.5004
Epoch 17/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.2029 - accurac
y: 0.5720 - val_loss: 1.4118 - val_accuracy: 0.5015
Epoch 18/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.1940 - accurac
y: 0.5762 - val_loss: 1.3831 - val_accuracy: 0.5152
Epoch 19/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.1850 - accurac
y: 0.5795 - val_loss: 1.4290 - val_accuracy: 0.4999
Epoch 20/20
1563/1563 [==============================] - 5s 3ms/step - loss: 1.1752 - accurac
y: 0.5809 - val_loss: 1.3786 - val_accuracy: 0.5199
```

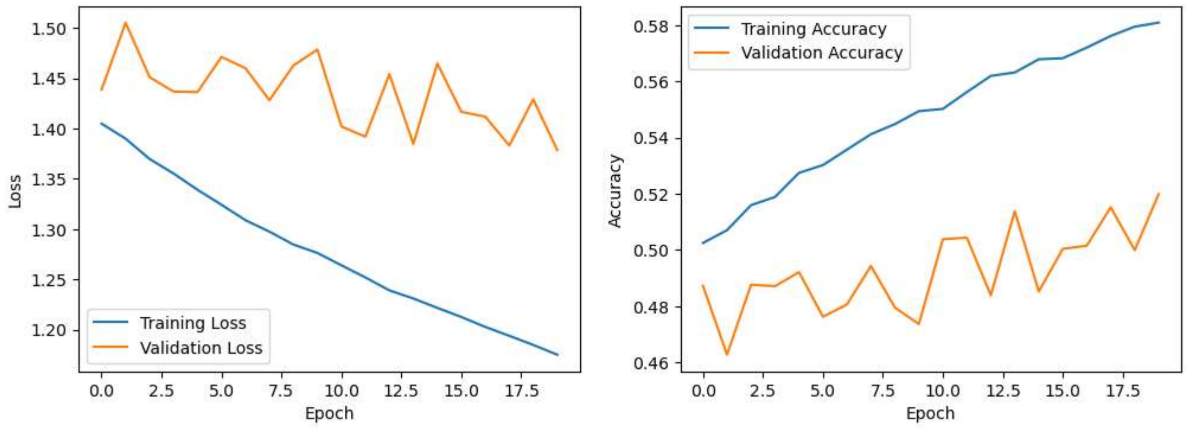
In [8]:

```
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
print(f"Test accuracy: {test_acc}")
```

```
313/313 - 0s - loss: 1.3786 - accuracy: 0.5199 - 448ms/epoch - 1ms/step
Test accuracy: 0.5199000239372253
```

In [9]:

```
# Plot training loss and accuracy
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```

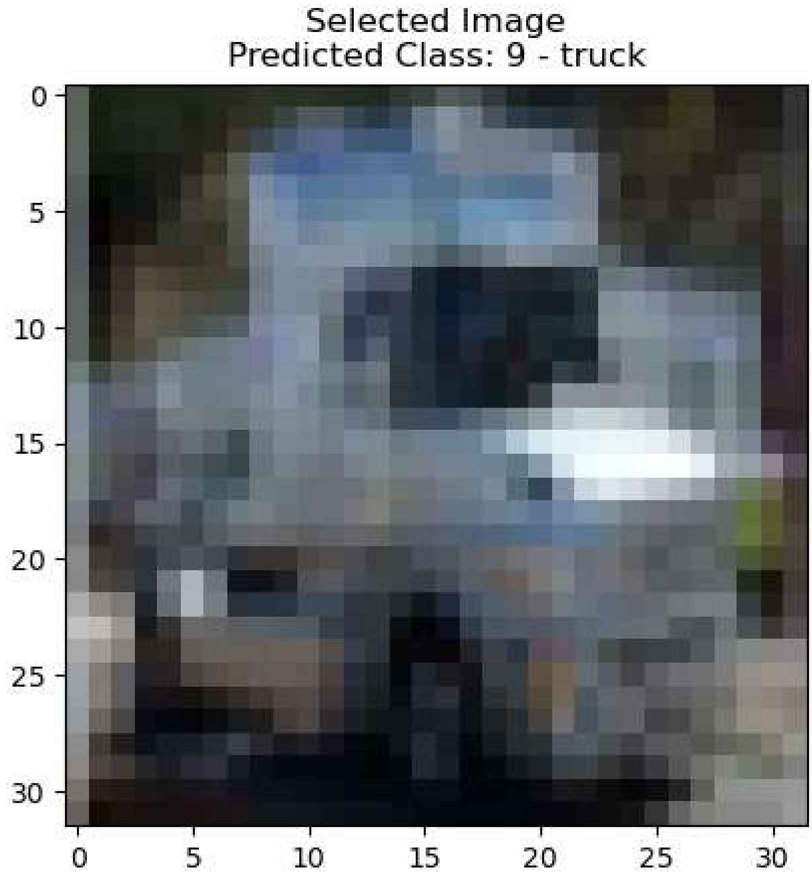


In [18]:

```
import numpy as np
import matplotlib.pyplot as plt

# Assuming you have a trained model named 'model'

# Load the CIFAR-10 dataset for testing
(test_images, test_labels) = datasets.cifar10.load_data()[1]

# Randomly select an image from the test dataset
index = np.random.randint(0, test_images.shape[0])
random_input = test_images[index:index + 1]

# Get the model's prediction for the selected image
prediction = model.predict(random_input)

# Find the class with the highest probability (argmax)
predicted_class = np.argmax(prediction)

# You can also have a mapping of class indices to class labels if available
class_labels = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "ho
predicted_label = class_labels[predicted_class]

# Display the selected image and the predicted class
plt.figure()
plt.imshow(random_input[0])
plt.title(f"Selected Image\nPredicted Class: {predicted_class} - {predicted_label}'
plt.show()
```

```
print("Predicted Class Probabilities:")
print(prediction)
```

```
1/1 [==============================] - 0s 19ms/step
```




```
Predicted Class Probabilities:
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```