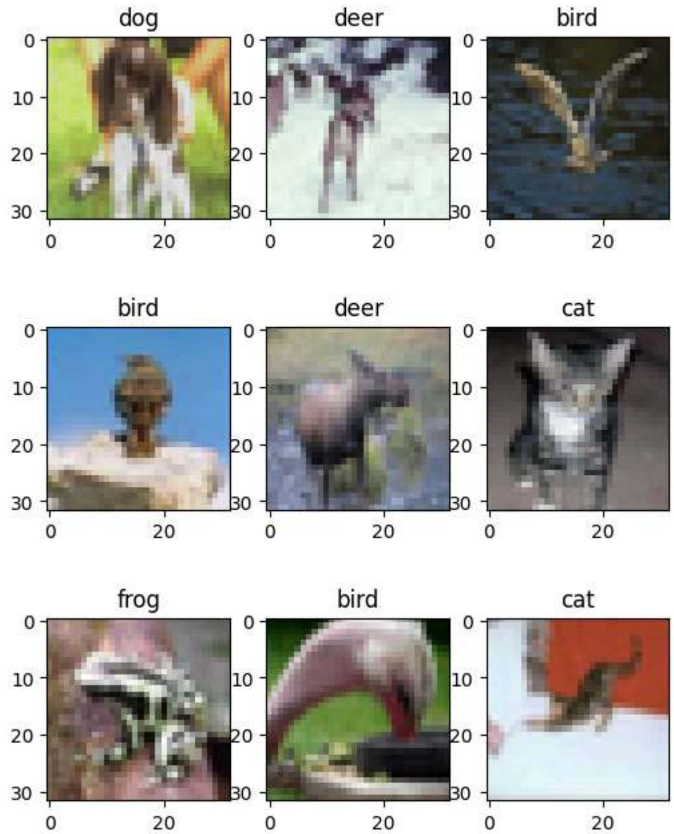
```
import os
import time
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
import tensorflow_datasets as tfds
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras import (models,layers,)
from matplotlib import pyplot as plt
import numpy as np
import random
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt6
```

```
cifar_10_labels = {
    0: 'airplane',
    1: 'automobile',
    2: 'bird',
    3: 'cat',
    4: 'deer',
    5: 'dog',
    6: 'frog',
    7: 'horse',
    8: 'ship',
    9: 'truck'
}
```

```
model_directory = 'models'

class RandomIntegers:
    def __init__(self):
        pass

    def generate(self, n, length):
        # Generate n unique random integers between 0 and length
        return random.sample(range(length), n)

# Load the CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
x_train = x_train / 255.0
x_test = x_test / 255.0
```

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

```
def display_images():
    random_integers = RandomIntegers().generate(9, len(x_train))
    plt.figure(figsize=(6, 8))
    for counter, i in enumerate(random_integers):
        plt.subplot(3, 3, counter + 1)
        plt.imshow(x_train[i])
        plt.title(cifar_10_labels[y_train[i][0]])
    plt.show()

display_images()
```



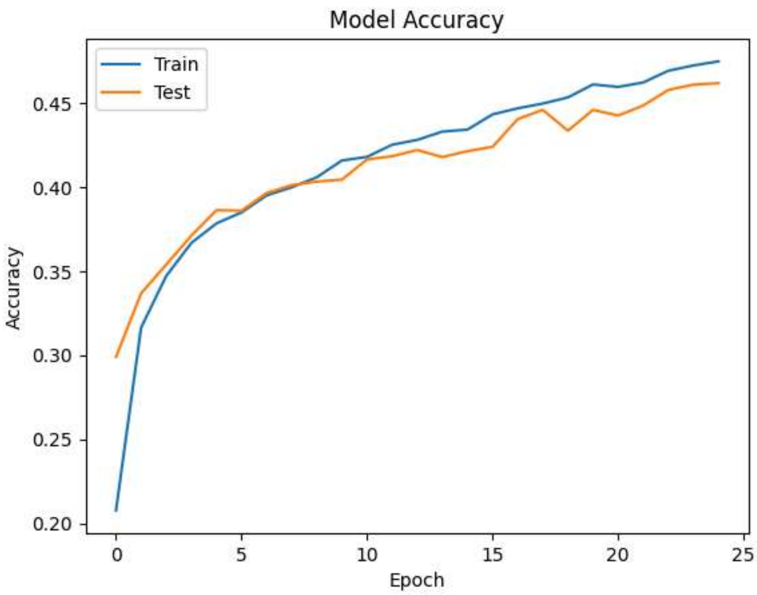

```
# Define models
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(32, 32, 3)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model_with_dropout = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(32, 32, 3)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
])

# Compile models
model.compile(
    optimizer=tf.keras.optimizers.Adam(0.001),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(),
    metrics=[tf.keras.metrics.SparseCategoricalAccuracy()]
)

model_with_dropout.compile(
    optimizer=tf.keras.optimizers.Adam(0.001),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(),
    metrics=[tf.keras.metrics.SparseCategoricalAccuracy()]
)

# Train models
start_time = time.time()
history = model.fit(
    x_train, y_train,
    epochs=25,
    batch_size=512,
    validation_data=(x_test, y_test)
)
end_time = time.time()
total_time = end_time - start_time
```

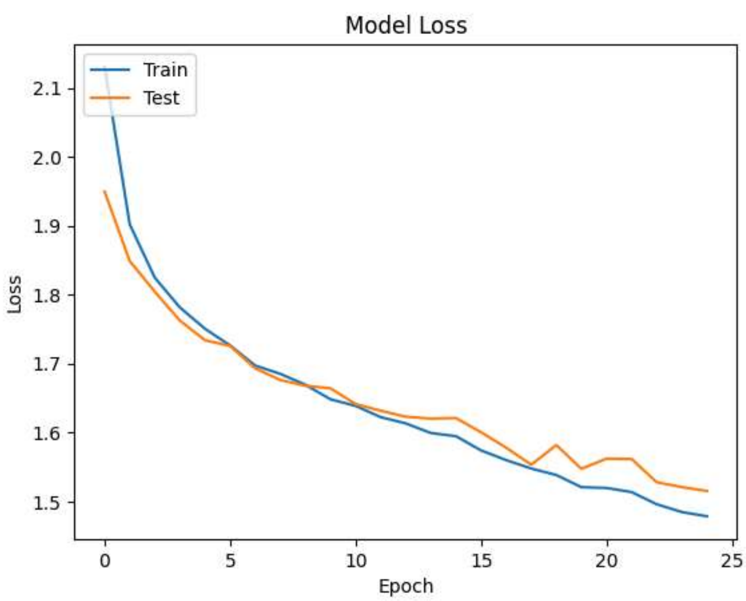
```
print("Time taken for training: ", total_time, " seconds")

start_time = time.time()
history_dropout = model_with_dropout.fit(
    x_train, y_train,
    epochs=25,
    batch_size=512,
    validation_data=(x_test, y_test)
)
end_time = time.time()
total_time = end_time - start_time
print("Time taken for training (Model with dropout): ", total_time, " seconds")
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input_shape`/`input_di
  super().__init__(**kwargs)
Epoch 1/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 29ms/step - loss: 2.1884 - sparse_categorical_accuracy: 0.1886 - val_loss: 1.9522 - val_sparse_categori
Epoch 2/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 36ms/step - loss: 1.9357 - sparse_categorical_accuracy: 0.2848 - val_loss: 1.8809 - val_sparse_categori
Epoch 3/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 31ms/step - loss: 1.8868 - sparse_categorical_accuracy: 0.3200 - val_loss: 1.8225 - val_sparse_categori
Epoch 4/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 25ms/step - loss: 1.8012 - sparse_categorical_accuracy: 0.3542 - val_loss: 1.7862 - val_sparse_categori
Epoch 5/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 25ms/step - loss: 1.7547 - sparse_categorical_accuracy: 0.3709 - val_loss: 1.7295 - val_sparse_categori
Epoch 6/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 25ms/step - loss: 1.7093 - sparse_categorical_accuracy: 0.3863 - val_loss: 1.6769 - val_sparse_categori
Epoch 7/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 39ms/step - loss: 1.6673 - sparse_categorical_accuracy: 0.4042 - val_loss: 1.6779 - val_sparse_categori
Epoch 8/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 25ms/step - loss: 1.6490 - sparse_categorical_accuracy: 0.4081 - val_loss: 1.6280 - val_sparse_categori
Epoch 9/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 25ms/step - loss: 1.6125 - sparse_categorical_accuracy: 0.4261 - val_loss: 1.6046 - val_sparse_categori
Epoch 10/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - loss: 1.6039 - sparse_categorical_accuracy: 0.4281 - val_loss: 1.5999 - val_sparse_categori
Epoch 11/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 33ms/step - loss: 1.5760 - sparse_categorical_accuracy: 0.4375 - val_loss: 1.6153 - val_sparse_categori
Epoch 12/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 25ms/step - loss: 1.5629 - sparse_categorical_accuracy: 0.4434 - val_loss: 1.5700 - val_sparse_categori
Epoch 13/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - loss: 1.5427 - sparse_categorical_accuracy: 0.4532 - val_loss: 1.5756 - val_sparse_categori
Epoch 14/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 25ms/step - loss: 1.5186 - sparse_categorical_accuracy: 0.4570 - val_loss: 1.5534 - val_sparse_categori
Epoch 15/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 36ms/step - loss: 1.4981 - sparse_categorical_accuracy: 0.4629 - val_loss: 1.5269 - val_sparse_categori
Epoch 16/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 25ms/step - loss: 1.4967 - sparse_categorical_accuracy: 0.4671 - val_loss: 1.5274 - val_sparse_categori
Epoch 17/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 25ms/step - loss: 1.4796 - sparse_categorical_accuracy: 0.4713 - val_loss: 1.5048 - val_sparse_categori
Epoch 18/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 25ms/step - loss: 1.4542 - sparse_categorical_accuracy: 0.4839 - val_loss: 1.5074 - val_sparse_categori
Epoch 19/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 33ms/step - loss: 1.4408 - sparse_categorical_accuracy: 0.4911 - val_loss: 1.4929 - val_sparse_categori
Epoch 20/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 34ms/step - loss: 1.4398 - sparse_categorical_accuracy: 0.4885 - val_loss: 1.5011 - val_sparse_categori
Epoch 21/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 25ms/step - loss: 1.4345 - sparse_categorical_accuracy: 0.4922 - val_loss: 1.5041 - val_sparse_categori
Epoch 22/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 25ms/step - loss: 1.4161 - sparse_categorical_accuracy: 0.4980 - val_loss: 1.4803 - val_sparse_categori
Epoch 23/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 29ms/step - loss: 1.3905 - sparse_categorical_accuracy: 0.5065 - val_loss: 1.5011 - val_sparse_categori
Epoch 24/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 36ms/step - loss: 1.3982 - sparse_categorical_accuracy: 0.4997 - val_loss: 1.4782 - val_sparse_categori
Epoch 25/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 25ms/step - loss: 1.4051 - sparse_categorical_accuracy: 0.5003 - val_loss: 1.4588 - val_sparse_categori
Time taken for training:  81.56557369232178  seconds
Epoch 1/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 32ms/step - loss: 2.3061 - sparse_categorical_accuracy: 0.1395 - val_loss: 1.9833 - val_sparse_categori
Epoch 2/25
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 41ms/step - loss: 2.0140 - sparse_categorical_accuracy: 0.2618 - val_loss: 1.8649 - val_sparse_categori
```

```
# Plot accuracy
plt.plot(history.history['sparse_categorical_accuracy'])
plt.plot(history.history['val_sparse_categorical_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

```
# Plot loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```






```
# Save models
if not os.path.exists(model_directory):
    os.makedirs(model_directory)

model_path = os.path.join(model_directory, "model_nn.h5")
model.save(model_path)

model_path_dropout = os.path.join(model_directory, "model_nn_dropout.h5")
model_with_dropout.save(model_path_dropout)

# Load saved models
loaded_model = tf.keras.models.load_model(model_path)
loaded_model_dropout = tf.keras.models.load_model(model_path_dropout)

loaded_model.summary()
loaded_model_dropout.summary()
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is consi
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is consi
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you t
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you t
```

Model: "sequential"

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 3072)	0
dense (Dense)	(None, 128)	393,344
dense_1 (Dense)	(None, 32)	4,128
dense_2 (Dense)	(None, 10)	330

Total params: 397,804 (1.52 MB)
Trainable params: 397,802 (1.52 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 2 (12.00 B)

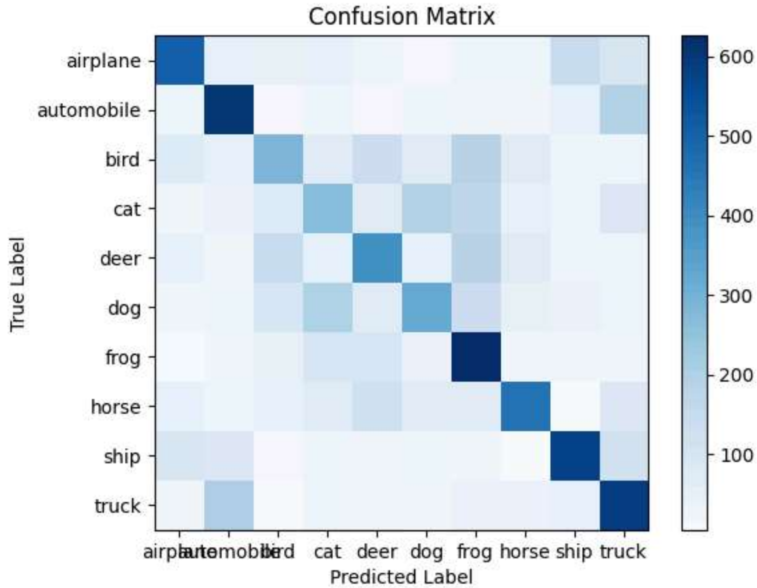
Model: "sequential_1"

Layer (type)	Output Shape	Param #
flatten_1 (Flatten)	(None, 3072)	0
dense_3 (Dense)	(None, 128)	393,344
dropout (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 32)	4,128
dropout_1 (Dropout)	(None, 32)	0
dense_5 (Dense)	(None, 10)	330

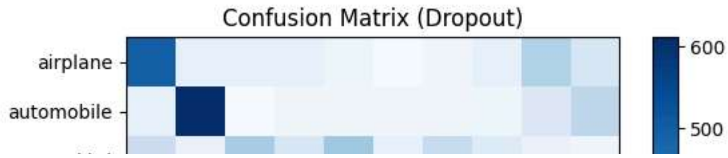
Total params: 397,804 (1.52 MB)
Trainable params: 397,802 (1.52 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 2 (12.00 B)

```
# Predict and evaluate
predicted_labels = np.argmax(loaded_model.predict(x_test), axis=-1)
predicted_labels_dropout = np.argmax(loaded_model_dropout.predict(x_test), axis=-1)
true_labels = y_test.flatten()

# Generate confusion matrices
cm = confusion_matrix(true_labels, predicted_labels)
cm_dropout = confusion_matrix(true_labels, predicted_labels_dropout)

# Plot confusion matrices
plt.imshow(cm, cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks(np.arange(10), list(cifar_10_labels.values()))
plt.yticks(np.arange(10), list(cifar_10_labels.values()))
plt.show()
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```




```
plt.imshow(cm_dropout, cmap=plt.cm.Blues)
plt.title('Confusion Matrix (Dropout)')
plt.colorbar()
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks(np.arange(10), list(cifar_10_labels.values()))
plt.yticks(np.arange(10), list(cifar_10_labels.values()))
plt.show()

# Generate classification reports
report = classification_report(true_labels, predicted_labels, target_names=list(cifar_10_labels.values()))
report_dropout = classification_report(true_labels, predicted_labels_dropout, target_names=list(cifar_10_labels.values()))

# Print classification reports
print("Classification Report (No Dropout):")
print(report)

print("Classification Report (With Dropout):")
print(report_dropout)
```


Confusion Matrix (Dropout)
airplane
automobile
600
500