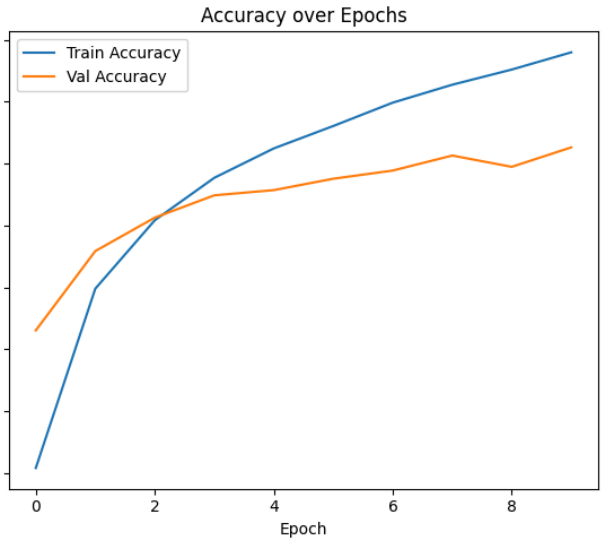
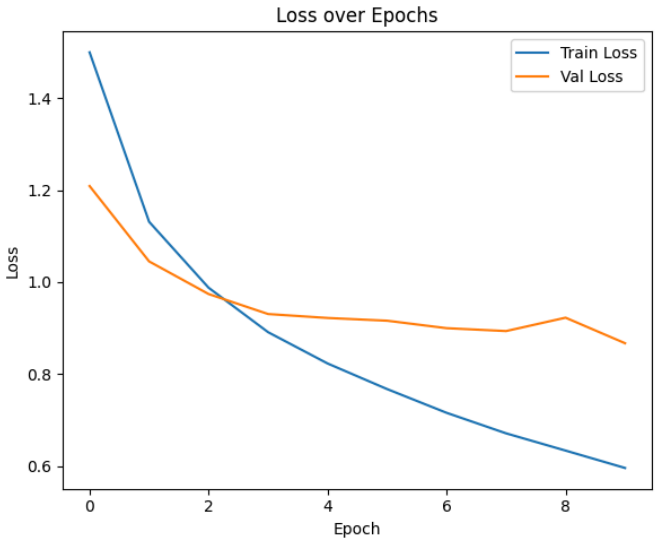
Gemini

```
 ✅ Step 1: Install TensorFlow (if needed)
pip install tensorflow --quiet

 ✅ Step 2: Import required libraries
mport tensorflow as tf
rom tensorflow.keras import layers, models
mport matplotlib.pyplot as plt
mport numpy as np
rom PIL import Image

 ✅ Step 3: Load CIFAR-10 dataset
x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_dat

 Normalize pixel values to [0, 1]
_train, x_test = x_train / 255.0, x_test / 255.0

 Class names for CIFAR-10
lass_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
              'dog', 'frog', 'horse', 'ship', 'truck']

 ✅ Step 4: Define CNN model
odel = models.Sequential([
   layers.Conv2D(32, (3,3), activation='relu', input_shape=(32,32,3)),
   layers.MaxPooling2D((2,2)),
   layers.Conv2D(64, (3,3), activation='relu'),
   layers.MaxPooling2D((2,2)),
   layers.Conv2D(64, (3,3), activation='relu'),
   layers.Flatten(),
   layers.Dense(64, activation='relu'),
   layers.Dense(10)  # Output layer (10 classes)
)

 ✅ Step 5: Compile the model
odel.compile(optimizer='adam',
             loss=tf.keras.losses.SparseCategoricalCrossentropy(from_log
             metrics=['accuracy'])

 ✅ Step 6: Train the model
istory = model.fit(x_train, y_train, epochs=10,
                   validation_data=(x_test, y_test))

 ✅ Step 7: Evaluate the model
est_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
rint(f"\n✅ Test Accuracy: {test_acc:.4f}")

 ✅ Step 8: Plot accuracy and loss
ef plot_training_history(history):
   plt.figure(figsize=(12,5))
```

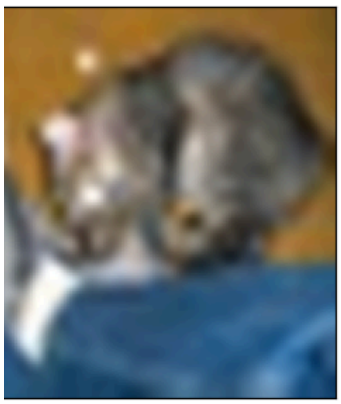
```
   # Accuracy
   plt.subplot(1,2,1)
   plt.plot(history.history['accuracy'], label='Train Accuracy')
   plt.plot(history.history['val_accuracy'], label='Val Accuracy')
   plt.title('Accuracy over Epochs')
   plt.xlabel('Epoch')
   plt.ylabel('Accuracy')
   plt.legend()

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

   plt.tight_layout()
   plt.show()

lot_training_history(history)

 ✅ Step 9: Predict and visualize test images
rob_model = tf.keras.Sequential([model, layers.Softmax()])
redictions = prob_model.predict(x_test)

ef show_predictions(images, labels, predictions, class_names):
   plt.figure(figsize=(10,10))
   for i in range(9):
       plt.subplot(3,3,i+1)
       plt.xticks([])
       plt.yticks([])
       plt.grid(False)

       # Resize image for better clarity
       img = Image.fromarray((images[i] * 255).astype(np.uint8))
       img_resized = img.resize((128, 128), Image.Resampling.LANCZOS)
       plt.imshow(img_resized)

       true_label = class_names[labels[i][0]]
       pred_label = class_names[np.argmax(predictions[i])]
       color = 'green' if true_label == pred_label else 'red'
       plt.xlabel(f"True: {true_label}\nPred: {pred_label}", color=color
   plt.tight_layout()
   plt.show()

how_predictions(x_test, y_test, predictions, class_names)
```

```
/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.
ber().__init__(activity_regularizer=activity_regularizer, **kwargs)
ı 1/10
/1563 ━━━━━━━━━━━━━━━━━━━━ 45s 28ms/step - accuracy: 0.3604 - loss: 1.730
ı 2/10
/1563 ━━━━━━━━━━━━━━━━━━━━ 84s 29ms/step - accuracy: 0.5835 - loss: 1.175
ı 3/10
/1563 ━━━━━━━━━━━━━━━━━━━━ 42s 27ms/step - accuracy: 0.6491 - loss: 1.001
ı 4/10
/1563 ━━━━━━━━━━━━━━━━━━━━ 45s 29ms/step - accuracy: 0.6886 - loss: 0.890
ı 5/10
/1563 ━━━━━━━━━━━━━━━━━━━━ 81s 28ms/step - accuracy: 0.7147 - loss: 0.819
ı 6/10
/1563 ━━━━━━━━━━━━━━━━━━━━ 80s 27ms/step - accuracy: 0.7325 - loss: 0.765
ı 7/10
/1563 ━━━━━━━━━━━━━━━━━━━━ 42s 27ms/step - accuracy: 0.7497 - loss: 0.710
ı 8/10
/1563 ━━━━━━━━━━━━━━━━━━━━ 81s 27ms/step - accuracy: 0.7666 - loss: 0.661
ı 9/10
/1563 ━━━━━━━━━━━━━━━━━━━━ 82s 27ms/step - accuracy: 0.7804 - loss: 0.622
ı 10/10
/1563 ━━━━━━━━━━━━━━━━━━━━ 87s 30ms/step - accuracy: 0.7964 - loss: 0.576
313 - 3s - 9ms/step - accuracy: 0.7131 - loss: 0.8671

est Accuracy: 0.7131
```





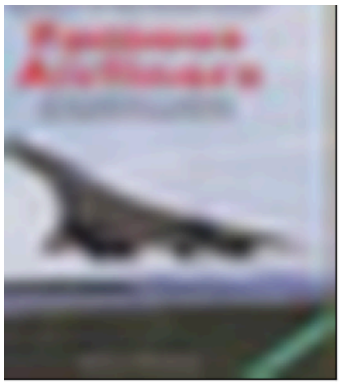

```
313 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step
```



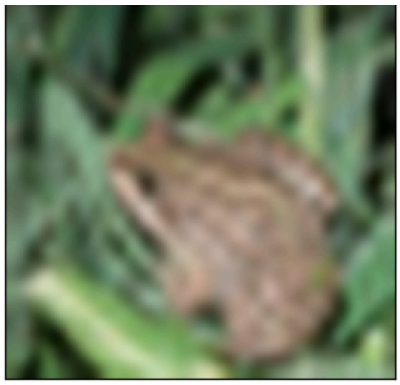
True: cat
Pred: cat

True: ship
Pred: ship

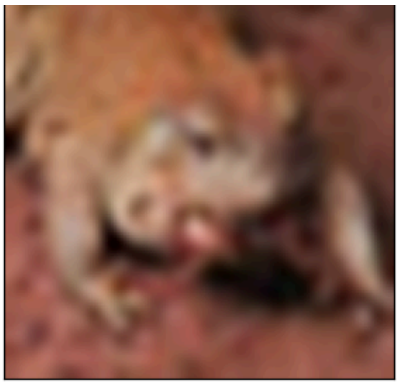
True: ship
Pred: airplane



True: airplane
Pred: airplane



True: frog
Pred: frog



True: frog
Pred: frog