```
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import SGD
from tensorflow.keras import regularizers
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split


#Loading the CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Preprocessing the data
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')

#Normalizing the greyscale intensities
x_train /= 255.0
x_test /= 255.0

#Converting the y values to categories
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)


#Splitting the dataset into training (70%), validation (10%), and test (20%) sets
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.3,
x_val, x_test, y_val, y_test = train_test_split(x_val, y_val, test_size=0.66, randc


x_train.shape, x_val.shape, x_test.shape, y_train.shape, y_val.shape, y_test.shape
```

```
((35000, 32, 32, 3),
 (5100, 32, 32, 3),
 (9900, 32, 32, 3),
 (35000, 10),
 (5100, 10),
 (9900, 10))
```

```
#Defining the CNN model
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel_regularizer=
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel_regularizer=
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel_regularizer=
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel_regularizer=
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.3))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel_regularizer
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel_regularizer
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dense(128, activation='relu', kernel_regularizer=regularizers.l2(0.001)))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))


#Setting the optimizer and compiling the model
model.compile(optimizer='SGD', loss='categorical_crossentropy', metrics=['accuracy'


batch_size = 128
epochs = 10
```

```
#Fitting the data to the CNN model
values = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validati
```

```
Epoch 1/10
274/274 [==============================] - 298s 1s/step - loss: 2.8963 - accu
Epoch 2/10
274/274 [==============================] - 292s 1s/step - loss: 2.8705 - accu
Epoch 3/10
274/274 [==============================] - 293s 1s/step - loss: 2.7706 - accu
Epoch 4/10
274/274 [==============================] - 293s 1s/step - loss: 2.6930 - accu
Epoch 5/10
274/274 [==============================] - 292s 1s/step - loss: 2.6508 - accu
Epoch 6/10
274/274 [==============================] - 298s 1s/step - loss: 2.6121 - accu
Epoch 7/10
274/274 [==============================] - 300s 1s/step - loss: 2.5846 - accu
Epoch 8/10
274/274 [==============================] - 298s 1s/step - loss: 2.5529 - accu
Epoch 9/10
274/274 [==============================] - 293s 1s/step - loss: 2.5208 - accu
Epoch 10/10
274/274 [==============================] - 291s 1s/step - loss: 2.4702 - accu
```

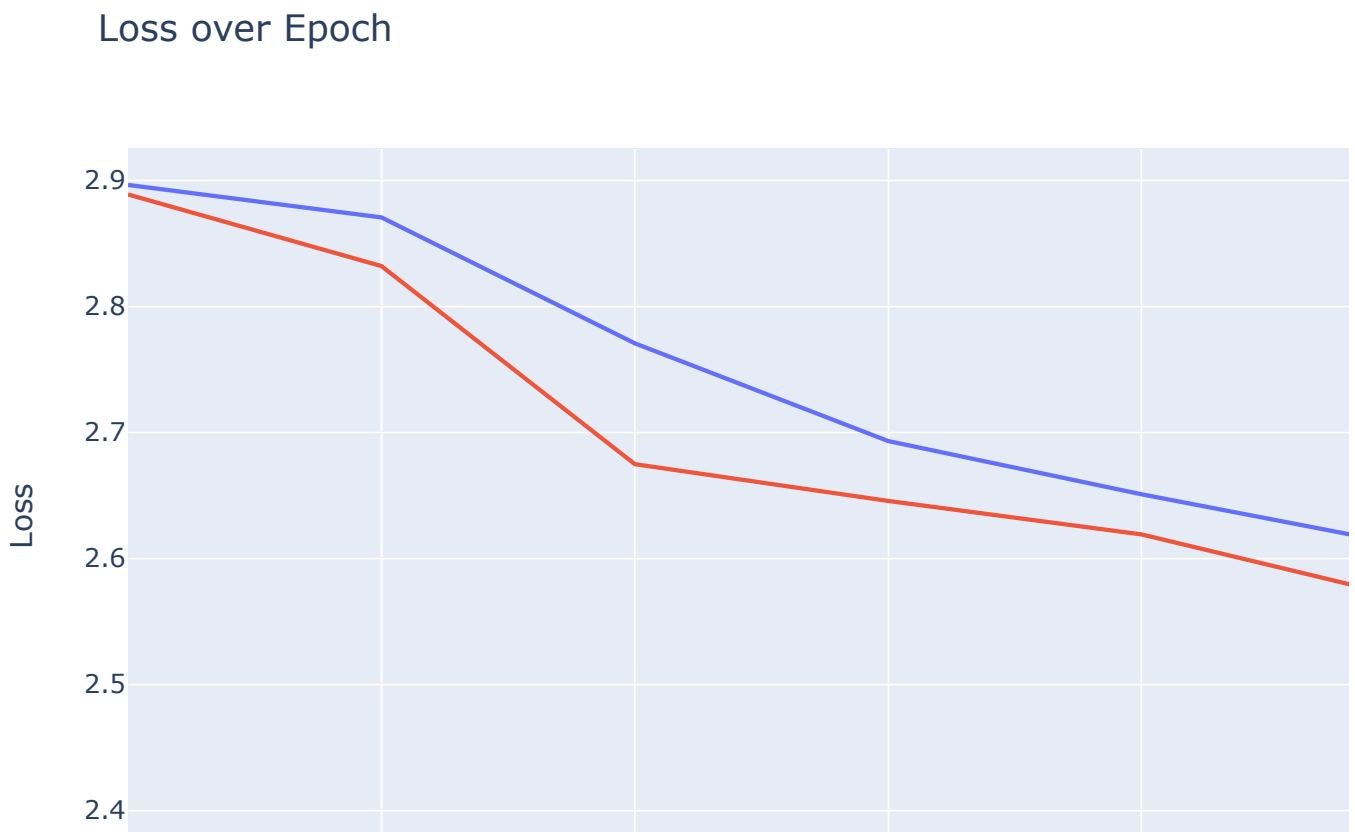
```
#Evaluating the model on the test set
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print(f"Test Loss: {test_loss:.4f}")
print(f"Test Accuracy: {test_accuracy:.4f}")
```

```
310/310 [==============================] - 20s 64ms/step - loss: 2.3583 - accu
Test Loss: 2.3583
Test Accuracy: 0.3208
```

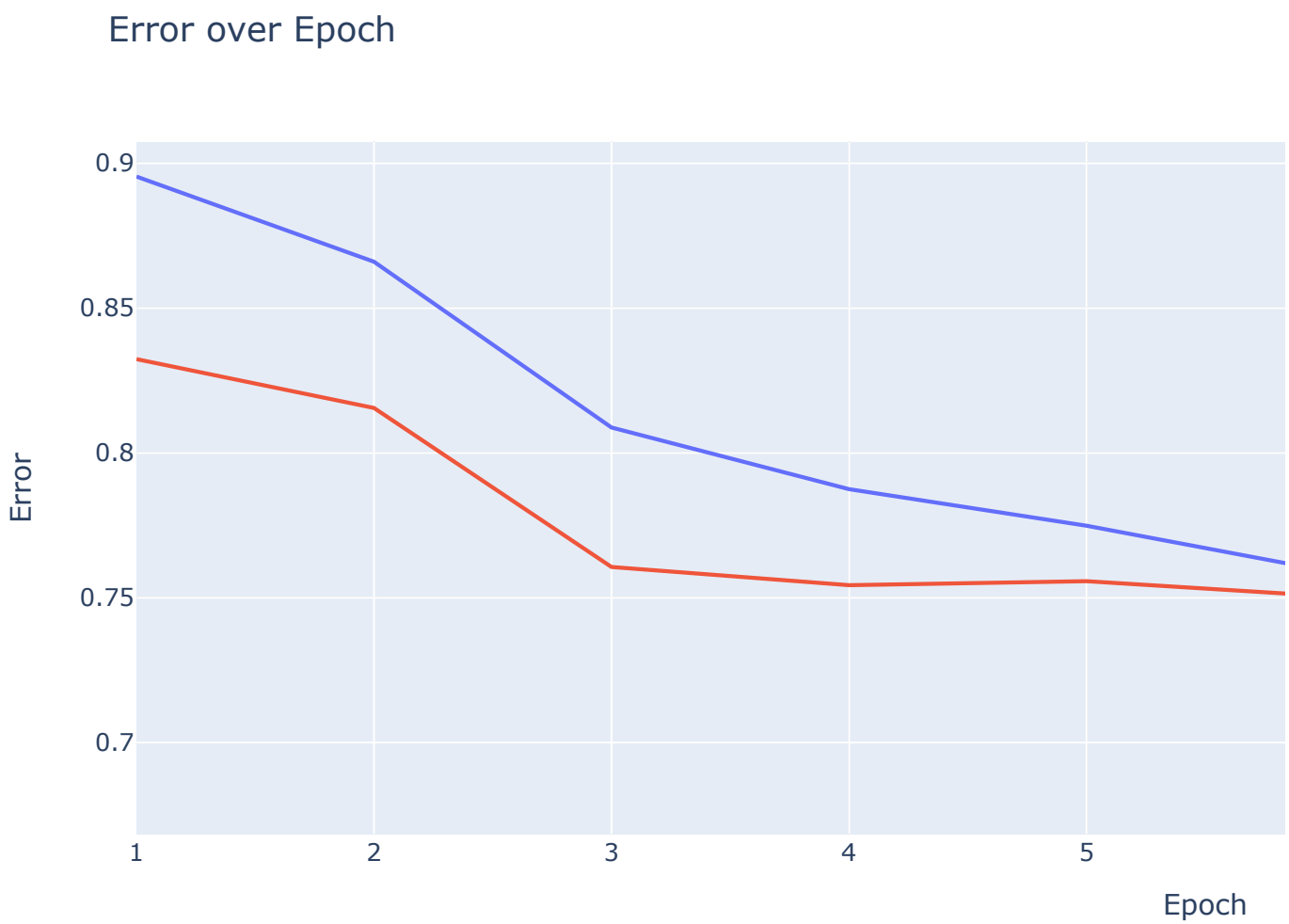
```
#Extracting loss and accuracy values
loss = values.history['loss']
val_loss = values.history['val_loss']
accuracy = values.history['accuracy']
val_accuracy = values.history['val_accuracy']
```

```
import plotly.graph_objects as go

epochs = range(1, len(loss) + 1)
epochsl = list(epochs)
```

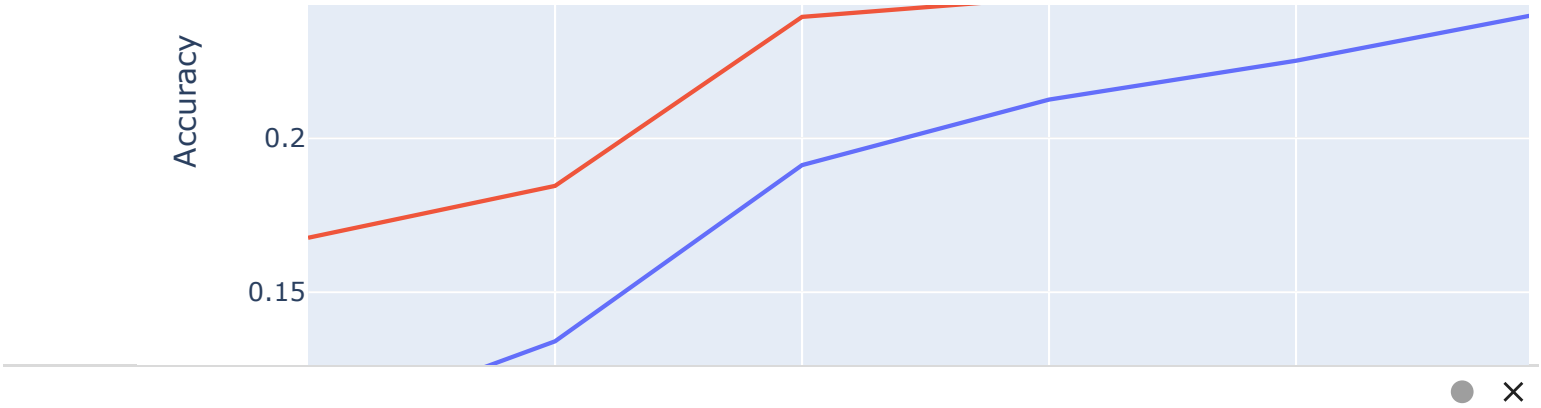
```
#Loss over Epoch plot
fig = go.Figure()
fig.add_trace(go.Scatter(x=epochsl, y=loss, mode='lines', name='Training Loss'))
fig.add_trace(go.Scatter(x=epochsl, y=val_loss, mode='lines', name='Validation Loss
fig.update_layout(title='Loss over Epoch', xaxis_title='Epoch', yaxis_title='Loss')
fig.show()


#Error over Epoch
fig = go.Figure()
fig.add_trace(go.Scatter(x=list(epochs), y=[1 - acc for acc in accuracy], mode='lir
fig.add_trace(go.Scatter(x=list(epochs), y=[1 - val_acc for val_acc in val_accuracy
fig.update_layout(title='Error over Epoch', xaxis_title='Epoch', yaxis_title='Error
fig.show()


#Accuracy over Epoch
fig = go.Figure()
fig.add_trace(go.Scatter(x=list(epochs), y=accuracy, mode='lines', name='Training A
fig.add_trace(go.Scatter(x=list(epochs), y=val_accuracy, mode='lines', name='Valida
fig.update_layout(title='Accuracy over Epoch', xaxis_title='Epoch', yaxis_title='Ac
fig.show()
```




1
2
3
4
5
Epoch


Error over Epoch
0.9
0.85
0.8
0.75
0.7
Error
1
2
3
4
5
Epoch


Accuracy over Epoch
0.3
0.25


Accuracy
0.2
0.15