```
import numpy as np
import os
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Model
from tensorflow.keras import utils, regularizers
from tensorflow.keras.layers import Input, Dense, BatchNormalization, Conv2D, MaxPooling2D, Flatten, Dropout
from keras.initializers import he_normal
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import classification_report
```

Load Data

```
(X_train, y_train), (X_test, y_test) = cifar10.load_data()


# Print the shape of the training and test datasets
print("Training data shape:", X_train.shape)
print("Training labels shape:", y_train.shape)
print("Test data shape:", X_test.shape)
print("Test labels shape:", y_test.shape)
print("=======================================")

# Get unique values in y_train and y_test
unique_values_train = np.unique(y_train)
unique_values_test = np.unique(y_test)
print("Unique values in y_train:", unique_values_train)
print("Unique values in y_test:", unique_values_test)
print("=======================================")

n_samples = X_train.shape[0]
n_features = X_train.shape[1:]
n_classes = len(unique_values_train)
print(f'Characteristics of train data:\nNumber of Samples: {n_samples}\nNumber of Features: {n_features}\nNumber of Classes: {n_classes}')
```

```
Training data shape: (50000, 32, 32, 3)
Training labels shape: (50000, 1)
Test data shape: (10000, 32, 32, 3)
Test labels shape: (10000, 1)
=======================================
Unique values in y_train: [0 1 2 3 4 5 6 7 8 9]
Unique values in y_test: [0 1 2 3 4 5 6 7 8 9]
=======================================
Characteristics of train data:
Number of Samples: 50000
Number of Features: (32, 32, 3)
Number of Classes: 10
```

```
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

# Print the corresponding class names for y_train
print("Class names for y_train:")
for label in np.unique(y_train):
    print(label, "-", class_names[label])
```

```
Class names for y_train:
0 - airplane
1 - automobile
2 - bird
3 - cat
4 - deer
5 - dog
6 - frog
7 - horse
8 - ship
9 - truck
```

```
X_train, X_test = X_train / 255.0, X_test / 255.0
```

```
y_train = tf.keras.utils.to_categorical(y_train, num_classes=n_classes)

y_test = tf.keras.utils.to_categorical(y_test, num_classes=n_classes)
```

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

1) Define your image classification model with convolutional layers (using thenn.Conv2d class from PyTorch).

```
def create_model():
  X = Input(shape = (32,32,3))
  x = Conv2D(32, kernel_size=(3, 3), activation='relu')(X)
  x = Conv2D(64, kernel_size=(3, 3), activation='relu')(x)
  x =  MaxPooling2D(pool_size=(2, 2))(x)

  x = Conv2D(128, kernel_size=(3, 3), activation='relu')(x)
  x = Conv2D(256, kernel_size=(3, 3), activation='relu')(x)
  x =  MaxPooling2D(pool_size=(2, 2))(x)

  x = Flatten()(x)
  x = Dense(128,activation = 'relu')(x)

  x =  Dense(10, activation="softmax", name='output')(x)

  model = Model(X,x)
  return model


model = create_model()
model.summary()
```

```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 32, 32, 3)]       0

 conv2d_8 (Conv2D)           (None, 30, 30, 32)        896

 conv2d_9 (Conv2D)           (None, 28, 28, 64)        18496

 max_pooling2d_4 (MaxPoolin  (None, 14, 14, 64)        0
 g2D)

 conv2d_10 (Conv2D)          (None, 12, 12, 128)       73856

 conv2d_11 (Conv2D)          (None, 10, 10, 256)       295168

 max_pooling2d_5 (MaxPoolin  (None, 5, 5, 256)         0
 g2D)

 flatten_2 (Flatten)         (None, 6400)              0

 dense_2 (Dense)             (None, 128)               819328

 output (Dense)              (None, 10)                1290

=================================================================
Total params: 1209034 (4.61 MB)
Trainable params: 1209034 (4.61 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

2) Choose between GPU (for faster training) or CPU to train your model.

```
import tensorflow as tf

# Check for available GPUs
physical_devices = tf.config.list_physical_devices('GPU')
if physical_devices:
  device = physical_devices[0]
  print("Using GPU:", device.name)

model = create_model()
metric = [tf.keras.metrics.CategoricalAccuracy()]
opt = tf.keras.optimizers.Adam()
loss = tf.keras.losses.CategoricalCrossentropy()
model.compile(loss=loss, optimizer=opt, metrics=metric)
batch_size = 256
n_epoch = 20
```

```
Using GPU: /physical_device:GPU:0
```

3) Plot the model's validation accuracy curve for each epoch

```
X_train.shape,y_train.shape
```

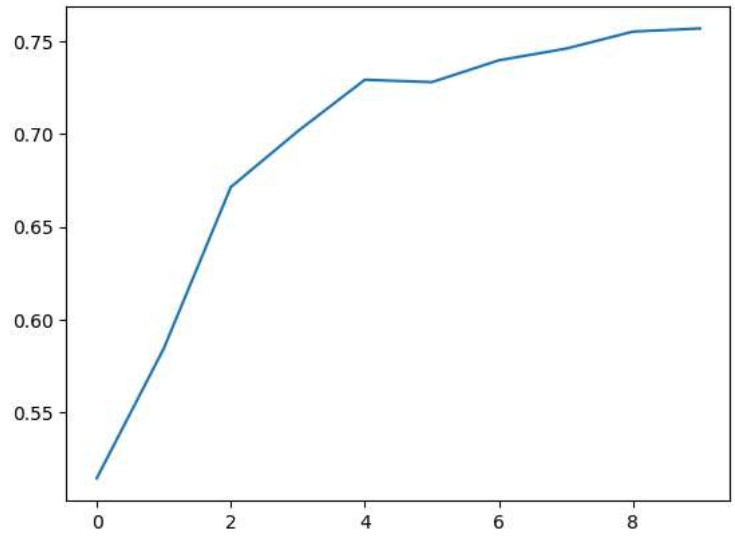
```
((50000, 32, 32, 3), (50000, 10))
```

```
history = model.fit(X_train, y_train,validation_data=(X_test,y_test), batch_size=batch_size,epochs = 10)
```

```
Epoch 1/10
196/196 [==============================] - 12s 33ms/step - loss: 1.6621 - categorical_accuracy: 0.3960 - val_loss: 1.3558 - val_categori
Epoch 2/10
196/196 [==============================] - 5s 27ms/step - loss: 1.2018 - categorical_accuracy: 0.5722 - val_loss: 1.1736 - val_categoric
Epoch 3/10
196/196 [==============================] - 6s 28ms/step - loss: 0.9894 - categorical_accuracy: 0.6540 - val_loss: 0.9470 - val_categoric
Epoch 4/10
196/196 [==============================] - 6s 29ms/step - loss: 0.8520 - categorical_accuracy: 0.7041 - val_loss: 0.8677 - val_categoric
Epoch 5/10
196/196 [==============================] - 6s 29ms/step - loss: 0.7562 - categorical_accuracy: 0.7371 - val_loss: 0.7782 - val_categoric
Epoch 6/10
196/196 [==============================] - 6s 29ms/step - loss: 0.6717 - categorical_accuracy: 0.7668 - val_loss: 0.7879 - val_categoric
Epoch 7/10
196/196 [==============================] - 6s 29ms/step - loss: 0.6079 - categorical_accuracy: 0.7898 - val_loss: 0.7515 - val_categoric
Epoch 8/10
196/196 [==============================] - 5s 28ms/step - loss: 0.5495 - categorical_accuracy: 0.8073 - val_loss: 0.7409 - val_categoric
Epoch 9/10
196/196 [==============================] - 6s 29ms/step - loss: 0.4927 - categorical_accuracy: 0.8279 - val_loss: 0.7220 - val_categoric
Epoch 10/10
196/196 [==============================] - 5s 27ms/step - loss: 0.4347 - categorical_accuracy: 0.8481 - val_loss: 0.7339 - val_categoric
```

```
plt.plot(history.history['val_categorical_accuracy'])
```

```
[<matplotlib.lines.Line2D at 0x7e6dfbe9dba0>]
```



Start coding or generate with AI.

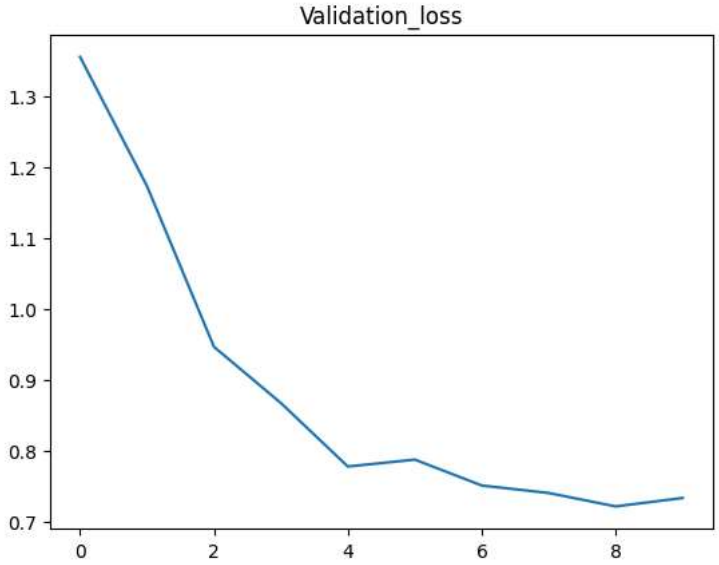
4) Plot training and validation losses curve to analyze trends.

```
plt.title('Training loss')
plt.plot(history.history['loss'])
```

```
[<matplotlib.lines.Line2D at 0x7e6dfbd5f790>]
```


```
plt.title("Validation_loss")
plt.plot(history.history['val_loss'])
```

```
[<matplotlib.lines.Line2D at 0x7e6dfbc007c0>]
```



Start coding or generate with AI.

5.Test your trained model's performance on three selected images.

```
X_test[0].shape
```

```
(32, 32, 3)
```

```
plt.figure(figsize=(10,10))
for i in range(3):
  plt.subplot(1,3,i+1)
  plt.imshow(X_test[i])
  plt.title(class_names[np.argmax(model.predict(np.expand_dims(X_test[i],axis = 0)))])
  plt.axis('off')
plt.show()
```

```
1/1 [==============================] - 0s 363ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 18ms/step
```

Start coding or generate with AI.

6.Save your model and reloading it.

```
model.save('model.h5')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `m
  saving_api.save_model(
```

```
from google.colab import files

files.download('/content/model.h5')
```

```
# Reload_model

from keras.models import load_model
model  = load_model('/content/model.h5')
```

Start coding or generate with AI.