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

import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.optimizers import SGD

from tensorflow.keras.layers import Input, Conv2D, Dense, Flatten,
Dropout
from tensorflow.keras.layers import GlobalMaxPooling2D, MaxPooling2D
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.models import Model

# Loading the test and train data
def load_dataset():
        (x_images, x_labels),(y_images, y_labels) =
cifar10.load_data()

(x_train, y_train), (x_test, y_test) = cifar10.load_data()
print(x_train.shape, y_train.shape, x_test.shape, y_test.shape)
```

```
(50000, 32, 32, 3) (50000, 1) (10000, 32, 32, 3) (10000, 1)
```

```python
# Reducing the pixel range between 0 and 1
x_train, x_test = x_train / 255.0, x_test / 255.0

# flatten the label values
y_train, y_test = y_train.flatten(), y_test.flatten()
```

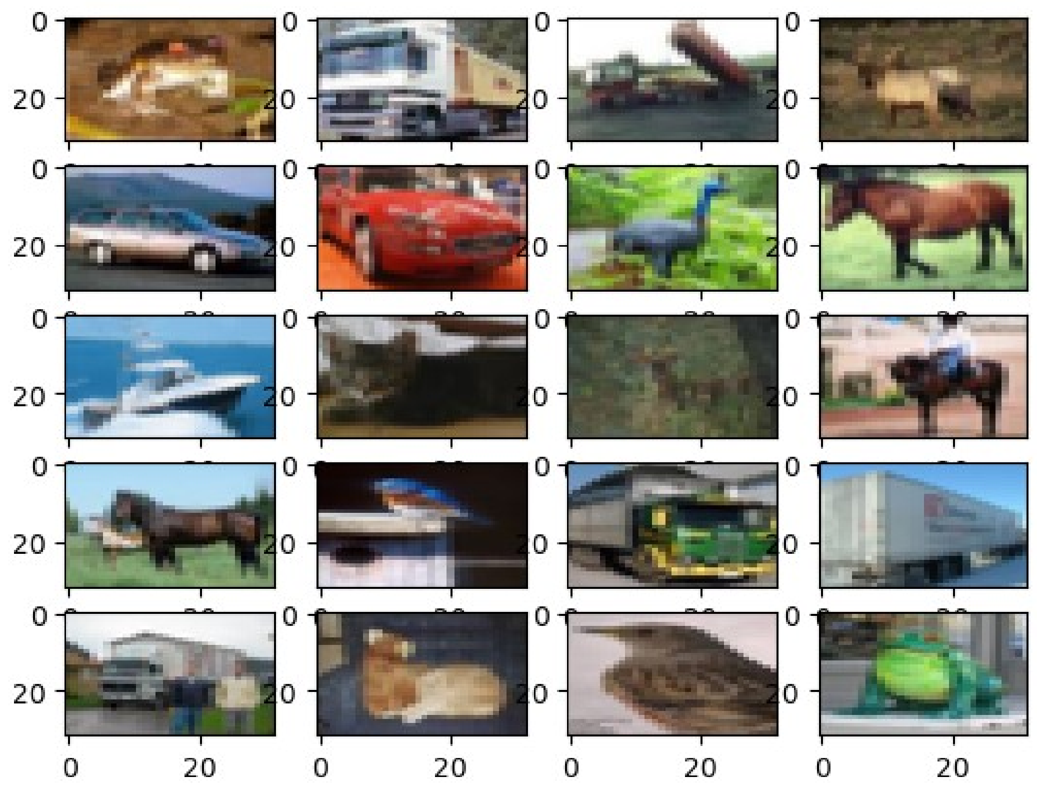
Here I want to see 20 images from the dataset so through visualization using the "subplot()" function from 'matplotlib' and looping over the first 20 images from the given training dataset portion

```python
# visualize the data by plotting images

fig, ax = plt.subplots(5, 4)
k = 0

for i in range(5):
    for j in range(4):
        ax[i][j].imshow(x_train[k], aspect='auto')
        k += 1

plt.show()
```



Defining the CNN(Convolutional Neural Network) to train the model

```
# print the number of classes
K = len(set(y_train))

# calculate total number of classes
# for output layer
print("number of classes:", K)

number of classes: 10

# Build the model using the functional API
# input layer

i = Input(shape=x_train[0].shape)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(i)
x = BatchNormalization()(x)

x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2))(x)

x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
```

```python
x = MaxPooling2D((2, 2))(x)

x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2))(x)

x = Flatten()(x)
x = Dropout(0.2)(x)

# Hidden layer
x = Dense(1024, activation='relu')(x)
x = Dropout(0.2)(x)

# last hidden layer i.e.. output layer
x = Dense(K, activation='softmax')(x)

model = Model(i, x)

# model description
model.summary()
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 32, 32, 3)]       0

 conv2d_6 (Conv2D)           (None, 32, 32, 32)        896

 batch_normalization_6 (Bat  (None, 32, 32, 32)        128
 chNormalization)

 conv2d_7 (Conv2D)           (None, 32, 32, 32)        9248

 batch_normalization_7 (Bat  (None, 32, 32, 32)        128
 chNormalization)

 max_pooling2d_3 (MaxPoolin  (None, 16, 16, 32)        0
 g2D)

 conv2d_8 (Conv2D)           (None, 16, 16, 64)        18496

 batch_normalization_8 (Bat  (None, 16, 16, 64)        256
 chNormalization)

 conv2d_9 (Conv2D)           (None, 16, 16, 64)        36928

 batch_normalization_9 (Bat  (None, 16, 16, 64)        256
 chNormalization)

 max_pooling2d_4 (MaxPoolin  (None, 8, 8, 64)          0
 g2D)

 conv2d_10 (Conv2D)          (None, 8, 8, 128)         73856

 batch_normalization_10 (Ba  (None, 8, 8, 128)         512
 tchNormalization)

 conv2d_11 (Conv2D)          (None, 8, 8, 128)         147584

 batch_normalization_11 (Ba  (None, 8, 8, 128)         512
 tchNormalization)

 max_pooling2d_5 (MaxPoolin  (None, 4, 4, 128)         0
 g2D)

 flatten_1 (Flatten)         (None, 2048)              0

 dropout_2 (Dropout)         (None, 2048)              0

 dense_2 (Dense)             (None, 1024)              2098176

 dropout_3 (Dropout)         (None, 1024)              0

 dense_3 (Dense)             (None, 10)                10250

=================================================================
Total params: 2397226 (9.14 MB)
Trainable params: 2396330 (9.14 MB)
Non-trainable params: 896 (3.50 KB)
_________________________________________________________________

# Compiling the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Fit
r = model.fit(
  x_train, y_train, validation_data=(x_test, y_test), epochs=25)

Epoch 1/25
1563/1563 [==============================] - 290s 184ms/step - loss:
1.3627 - accuracy: 0.5266 - val_loss: 1.4927 - val_accuracy: 0.5227
Epoch 2/25
1563/1563 [==============================] - 287s 183ms/step - loss:
0.8708 - accuracy: 0.6987 - val_loss: 1.2719 - val_accuracy: 0.5709
Epoch 3/25
```

```
1563/1563 [==============================] - 289s 185ms/step - loss:
0.7099 - accuracy: 0.7539 - val_loss: 1.1931 - val_accuracy: 0.6529
Epoch 4/25
1563/1563 [==============================] - 284s 181ms/step - loss:
0.6029 - accuracy: 0.7932 - val_loss: 1.0839 - val_accuracy: 0.6548
Epoch 5/25
1563/1563 [==============================] - 286s 183ms/step - loss:
0.5105 - accuracy: 0.8245 - val_loss: 0.9107 - val_accuracy: 0.7267
Epoch 6/25
1563/1563 [==============================] - 281s 180ms/step - loss:
0.4336 - accuracy: 0.8510 - val_loss: 2.8016 - val_accuracy: 0.4136
Epoch 7/25
1563/1563 [==============================] - 380s 243ms/step - loss:
0.3672 - accuracy: 0.8726 - val_loss: 0.7294 - val_accuracy: 0.7703
Epoch 8/25
1563/1563 [==============================] - 352s 225ms/step - loss:
0.3105 - accuracy: 0.8924 - val_loss: 1.3383 - val_accuracy: 0.6253
Epoch 9/25
1563/1563 [==============================] - 355s 227ms/step - loss:
0.2611 - accuracy: 0.9089 - val_loss: 0.9456 - val_accuracy: 0.7333
Epoch 10/25
1563/1563 [==============================] - 350s 224ms/step - loss:
0.2287 - accuracy: 0.9220 - val_loss: 0.9363 - val_accuracy: 0.7608
Epoch 11/25
1563/1563 [==============================] - 358s 229ms/step - loss:
0.2022 - accuracy: 0.9300 - val_loss: 1.8841 - val_accuracy: 0.6241
Epoch 12/25
1563/1563 [==============================] - 362s 231ms/step - loss:
0.1801 - accuracy: 0.9374 - val_loss: 1.2133 - val_accuracy: 0.7224
Epoch 13/25
1563/1563 [==============================] - 380s 243ms/step - loss:
0.1639 - accuracy: 0.9448 - val_loss: 9.1491 - val_accuracy: 0.2523
Epoch 14/25
1563/1563 [==============================] - 363s 232ms/step - loss:
0.1500 - accuracy: 0.9497 - val_loss: 22.5144 - val_accuracy: 0.1037
Epoch 15/25
1563/1563 [==============================] - 340s 217ms/step - loss:
0.1476 - accuracy: 0.9509 - val_loss: 5.9450 - val_accuracy: 0.1043
Epoch 16/25
1563/1563 [==============================] - 351s 224ms/step - loss:
0.1275 - accuracy: 0.9568 - val_loss: 14.1067 - val_accuracy: 0.1877
Epoch 17/25
1563/1563 [==============================] - 344s 220ms/step - loss:
0.1196 - accuracy: 0.9603 - val_loss: 13.4314 - val_accuracy: 0.1412
Epoch 18/25
1563/1563 [==============================] - 349s 224ms/step - loss:
0.1100 - accuracy: 0.9641 - val_loss: 5.1142 - val_accuracy: 0.2521
Epoch 19/25
1563/1563 [==============================] - 339s 217ms/step - loss:
```

```
0.1060 - accuracy: 0.9645 - val_loss: 11.1176 - val_accuracy: 0.2229
Epoch 20/25
1563/1563 [==============================] - 356s 228ms/step - loss:
0.1029 - accuracy: 0.9662 - val_loss: 2.4099 - val_accuracy: 0.6178
Epoch 21/25
1563/1563 [==============================] - 339s 217ms/step - loss:
0.0943 - accuracy: 0.9686 - val_loss: 8.8886 - val_accuracy: 0.2418
Epoch 22/25
1563/1563 [==============================] - 333s 213ms/step - loss:
0.0892 - accuracy: 0.9712 - val_loss: 6.8286 - val_accuracy: 0.1435
Epoch 23/25
1563/1563 [==============================] - 333s 213ms/step - loss:
0.0862 - accuracy: 0.9718 - val_loss: 3.8009 - val_accuracy: 0.4075
Epoch 24/25
1563/1563 [==============================] - 327s 209ms/step - loss:
0.0817 - accuracy: 0.9727 - val_loss: 2.0587 - val_accuracy: 0.6132
Epoch 25/25
1563/1563 [==============================] - 345s 220ms/step - loss:
0.0821 - accuracy: 0.9730 - val_loss: 3.3185 - val_accuracy: 0.3886

batch_size = 32
data_generator = tf.keras.preprocessing.image.ImageDataGenerator(
  width_shift_range=0.1, height_shift_range=0.1, horizontal_flip=True)

train_generator = data_generator.flow(x_train, y_train, batch_size)
steps_per_epoch = x_train.shape[0] // batch_size

r = model.fit(train_generator, validation_data=(x_test, y_test),
              steps_per_epoch=steps_per_epoch, epochs=25)

Epoch 1/25
1562/1562 [==============================] - 382s 243ms/step - loss:
0.6906 - accuracy: 0.7673 - val_loss: 0.5881 - val_accuracy: 0.8041
Epoch 2/25
1562/1562 [==============================] - 403s 258ms/step - loss:
0.6235 - accuracy: 0.7894 - val_loss: 0.5650 - val_accuracy: 0.8090
Epoch 3/25
1562/1562 [==============================] - 398s 255ms/step - loss:
0.5820 - accuracy: 0.8017 - val_loss: 0.6103 - val_accuracy: 0.7965
Epoch 4/25
1562/1562 [==============================] - 417s 267ms/step - loss:
0.5456 - accuracy: 0.8140 - val_loss: 0.7298 - val_accuracy: 0.7691
Epoch 5/25
1562/1562 [==============================] - 400s 256ms/step - loss:
0.5195 - accuracy: 0.8236 - val_loss: 0.5404 - val_accuracy: 0.8181
Epoch 6/25
1562/1562 [==============================] - 373s 239ms/step - loss:
0.4956 - accuracy: 0.8302 - val_loss: 0.5308 - val_accuracy: 0.8259
Epoch 7/25
1562/1562 [==============================] - 369s 236ms/step - loss:
```

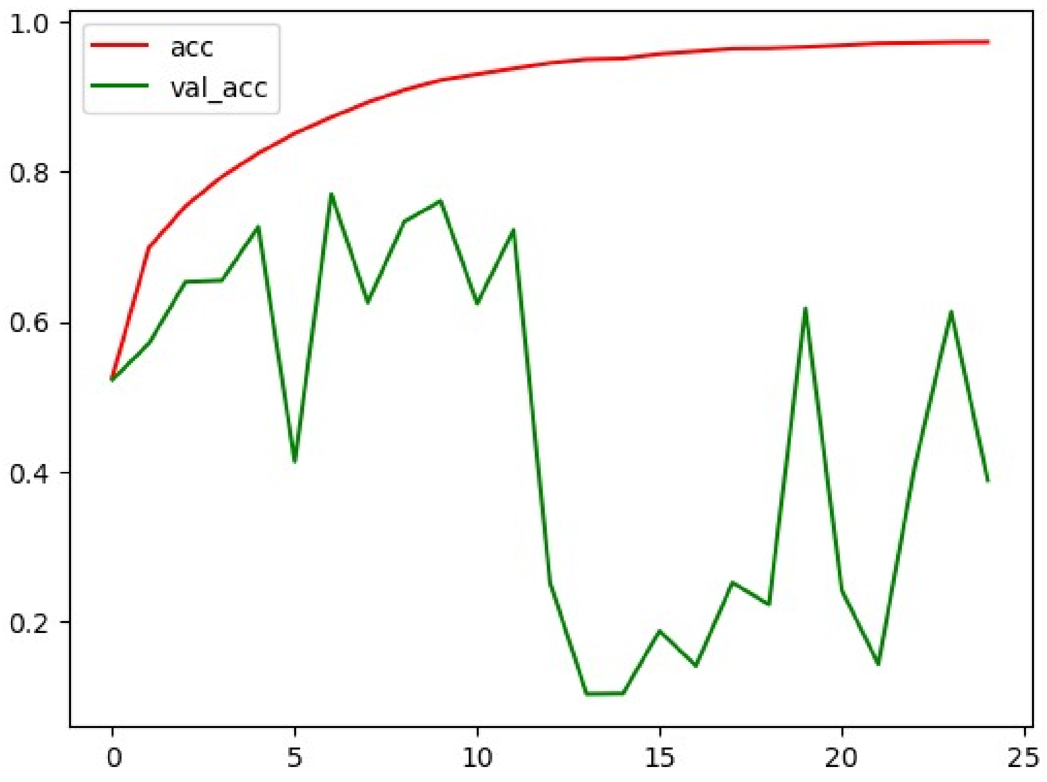
```
0.4673 - accuracy: 0.8390 - val_loss: 0.5169 - val_accuracy: 0.8286
Epoch 8/25
1562/1562 [==============================] - 387s 248ms/step - loss:
0.4524 - accuracy: 0.8442 - val_loss: 0.4672 - val_accuracy: 0.8456
Epoch 9/25
1562/1562 [==============================] - 377s 241ms/step - loss:
0.4307 - accuracy: 0.8529 - val_loss: 0.4706 - val_accuracy: 0.8433
Epoch 10/25
1562/1562 [==============================] - 306s 196ms/step - loss:
0.4141 - accuracy: 0.8562 - val_loss: 0.5428 - val_accuracy: 0.8264
Epoch 11/25
1562/1562 [==============================] - 334s 214ms/step - loss:
0.3932 - accuracy: 0.8629 - val_loss: 0.4455 - val_accuracy: 0.8525
Epoch 12/25
1562/1562 [==============================] - 345s 221ms/step - loss:
0.3835 - accuracy: 0.8687 - val_loss: 0.4474 - val_accuracy: 0.8524
Epoch 13/25
1562/1562 [==============================] - 328s 210ms/step - loss:
0.3704 - accuracy: 0.8722 - val_loss: 0.4297 - val_accuracy: 0.8606
Epoch 14/25
1562/1562 [==============================] - 333s 213ms/step - loss:
0.3568 - accuracy: 0.8779 - val_loss: 0.4749 - val_accuracy: 0.8483
Epoch 15/25
1562/1562 [==============================] - 337s 216ms/step - loss:
0.3481 - accuracy: 0.8794 - val_loss: 0.4440 - val_accuracy: 0.8565
Epoch 16/25
1562/1562 [==============================] - 355s 227ms/step - loss:
0.3294 - accuracy: 0.8857 - val_loss: 0.4771 - val_accuracy: 0.8457
Epoch 17/25
1562/1562 [==============================] - 349s 224ms/step - loss:
0.3257 - accuracy: 0.8876 - val_loss: 0.4767 - val_accuracy: 0.8503
Epoch 18/25
1562/1562 [==============================] - 369s 236ms/step - loss:
0.3167 - accuracy: 0.8910 - val_loss: 0.4242 - val_accuracy: 0.8604
Epoch 19/25
1562/1562 [==============================] - 345s 221ms/step - loss:
0.3112 - accuracy: 0.8918 - val_loss: 0.4419 - val_accuracy: 0.8582
Epoch 20/25
1562/1562 [==============================] - 365s 234ms/step - loss:
0.2968 - accuracy: 0.8969 - val_loss: 0.4293 - val_accuracy: 0.8619
Epoch 21/25
1562/1562 [==============================] - 356s 228ms/step - loss:
0.2938 - accuracy: 0.8975 - val_loss: 0.4246 - val_accuracy: 0.8667
Epoch 22/25
1562/1562 [==============================] - 335s 214ms/step - loss:
0.2861 - accuracy: 0.8998 - val_loss: 0.4573 - val_accuracy: 0.8594
Epoch 23/25
1562/1562 [==============================] - 342s 219ms/step - loss:
0.2791 - accuracy: 0.9035 - val_loss: 0.5191 - val_accuracy: 0.8499
```

```
Epoch 24/25
1562/1562 [==============================] - 336s 215ms/step - loss:
0.2707 - accuracy: 0.9050 - val_loss: 0.4807 - val_accuracy: 0.8436
Epoch 25/25
1562/1562 [==============================] - 311s 199ms/step - loss:
0.2689 - accuracy: 0.9081 - val_loss: 0.4448 - val_accuracy: 0.8609
```

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

# Plot accuracy per iteration
plt.plot(r.history['accuracy'], label='acc', color='red')
plt.plot(r.history['val_accuracy'], label='val_acc', color='green')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1bb9a031750>
```


acc
val_acc