PARINATI GAUR 20BAI10149

Importing the required libraries

```
# Library for plotting the images·and·the·loss·function
import·matplotlib.pyplot·as·plt

#·We·import·the·data·set·from·tensorflow and build the model there
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
```

Downloading the dataset and normalizing the pixel values

```
# Download the data set
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()

# Normalize pixel values between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 11s 0us/step
```

Defining the image classes

```
# Define the 10 image classes
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

# Show the first 10 images
plt.figure(figsize=(10,10))
for i in range(10):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i])
    # Die CIFAR Labels sind Arrays, deshalb benötigen wir den extra Index
```

```
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()
```

Building the Convolution Neural Network

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
```

Model Summary

```
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
```

```
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 4, 4, 64)          36928     
                                                                 
 flatten (Flatten)           (None, 1024)              0         
                                                                 
 dense (Dense)               (None, 64)                65600     
                                                                 
 dense_1 (Dense)             (None, 10)                650       
                                                                 
=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

Fitting the Model: Train, Test and Validation

```
model.compile(optimizer='adam',
             loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
             metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=20,
                    validation_data=(test_images, test_labels))
```

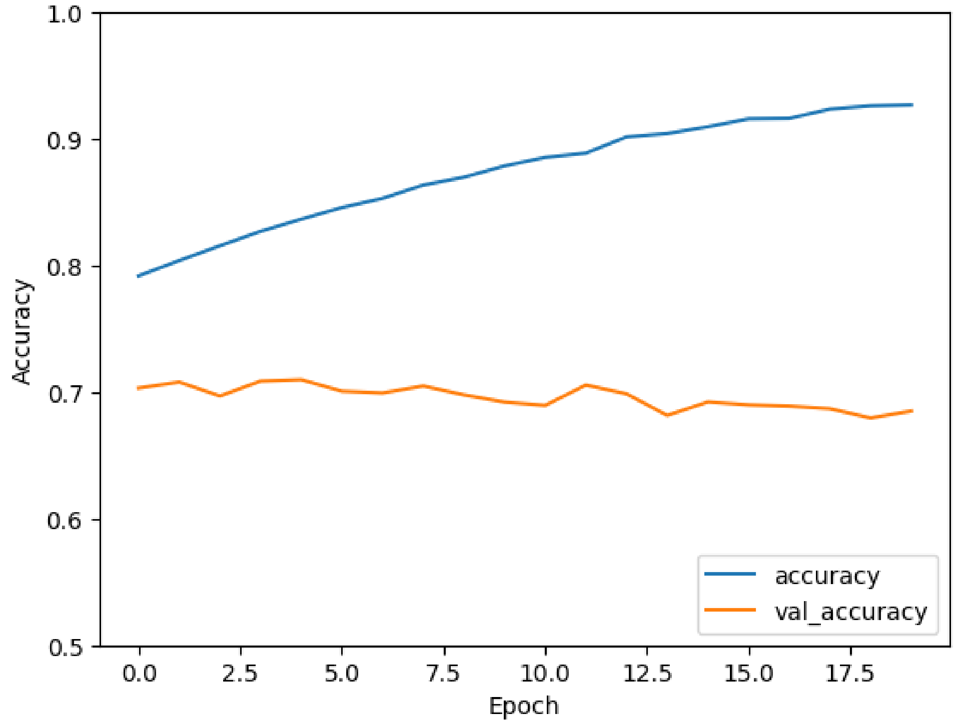
```
Epoch 1/20
1563/1563 [==============================] - 84s 52ms/step - loss: 0.5880 - accuracy: 0.7919 - val_loss: 0.9096 - val_accuracy: 0.7032
Epoch 2/20
1563/1563 [==============================] - 79s 51ms/step - loss: 0.5529 - accuracy: 0.8039 - val_loss: 0.9222 - val_accuracy: 0.7079
Epoch 3/20
1563/1563 [==============================] - 79s 50ms/step - loss: 0.5193 - accuracy: 0.8156 - val_loss: 0.9450 - val_accuracy: 0.6969
Epoch 4/20
1563/1563 [==============================] - 80s 51ms/step - loss: 0.4882 - accuracy: 0.8270 - val_loss: 0.9419 - val_accuracy: 0.7087
Epoch 5/20
1563/1563 [==============================] - 80s 51ms/step - loss: 0.4601 - accuracy: 0.8366 - val_loss: 0.9800 - val_accuracy: 0.7098
```

```
Epoch 6/20
1563/1563 [==============================] - 79s 51ms/step - loss: 0.4312 - accuracy: 0.8458 - val_loss: 1.0223 - val_accuracy: 0.7007
Epoch 7/20
1563/1563 [==============================] - 80s 51ms/step - loss: 0.4083 - accuracy: 0.8531 - val_loss: 1.0545 - val_accuracy: 0.6993
Epoch 8/20
1563/1563 [==============================] - 79s 51ms/step - loss: 0.3809 - accuracy: 0.8636 - val_loss: 1.1024 - val_accuracy: 0.7048
Epoch 9/20
1563/1563 [==============================] - 82s 53ms/step - loss: 0.3626 - accuracy: 0.8698 - val_loss: 1.1945 - val_accuracy: 0.6978
Epoch 10/20
1563/1563 [==============================] - 80s 51ms/step - loss: 0.3386 - accuracy: 0.8787 - val_loss: 1.2077 - val_accuracy: 0.6921
Epoch 11/20
1563/1563 [==============================] - 84s 54ms/step - loss: 0.3160 - accuracy: 0.8854 - val_loss: 1.2879 - val_accuracy: 0.6895
Epoch 12/20
1563/1563 [==============================] - 80s 51ms/step - loss: 0.3052 - accuracy: 0.8888 - val_loss: 1.2681 - val_accuracy: 0.7056
Epoch 13/20
1563/1563 [==============================] - 79s 51ms/step - loss: 0.2744 - accuracy: 0.9017 - val_loss: 1.3194 - val_accuracy: 0.6986
Epoch 14/20
1563/1563 [==============================] - 80s 51ms/step - loss: 0.2671 - accuracy: 0.9043 - val_loss: 1.4416 - val_accuracy: 0.6816
Epoch 15/20
1563/1563 [==============================] - 79s 51ms/step - loss: 0.2505 - accuracy: 0.9096 - val_loss: 1.4468 - val_accuracy: 0.6922
Epoch 16/20
1563/1563 [==============================] - 90s 57ms/step - loss: 0.2359 - accuracy: 0.9159 - val_loss: 1.5394 - val_accuracy: 0.6899
Epoch 17/20
1563/1563 [==============================] - 82s 52ms/step - loss: 0.2347 - accuracy: 0.9162 - val_loss: 1.6164 - val_accuracy: 0.6889
Epoch 18/20
1563/1563 [==============================] - 80s 51ms/step - loss: 0.2141 - accuracy: 0.9236 - val_loss: 1.6543 - val_accuracy: 0.6869
Epoch 19/20
1563/1563 [==============================] - 91s 58ms/step - loss: 0.2050 - accuracy: 0.9262 - val_loss: 1.7266 - val_accuracy: 0.6796
Epoch 20/20
1563/1563 [==============================] - 82s 52ms/step - loss: 0.2003 - accuracy: 0.9269 - val_loss: 1.8013 - val_accuracy: 0.6850
```

Evaluating the Model

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')
```

```
<matplotlib.legend.Legend at 0x7fd9c853cc10>
```



✓ 0s completed at 11:27